Machine Learning-Driven Chemical System Predictions and Microscopy Image Analysis For Energy Applications

A Dissertation Presented for the

Doctor of Philosophy

Degree

The University of Tennessee, Knoxville

Nicholas R. Furth

May 2026

Copyright © 2026 by Nicholas R. Furth

All rights reserved.

ACKNOWLEDGEMENTS

This dissertation would not have been possible without the support and guidance of many individuals. I am especially grateful to my dissertation advisor, Dr. Thomas A. Zawodzinski, for his mentorship and support throughout my years at the University of Tennessee (UT). I also sincerely thank my committee members, Dr. Manolis Doxastakis, Dr. Adam E. Imel, and Dr. Sergei V. Kalinin, for their valuable input and support.

I appreciate the camaraderie and assistance of my lab colleagues, including senior research staff, postdoctoral researchers, and fellow graduate students, as well as the staff and students of the Bredesen Center, particularly Allie Burns, Rachel Roberts, and Dr. Brynn Voy, for their continued advisement and support during my time at UT. I am thankful to the UT Science Alliance and the University of Tennessee-Oak Ridge Innovation Institute (UT-ORII) for funding through the Graduate Advancement and Training Education (GATE) fellowship, and to Dr. Shawn Campagna for his early support through the Student Mentoring and Research Training (SMaRT) program and for allowing me to mentor two students in the SMaRT program whom I would also like to thank; Daniel Joy and Ya'Donnis Hardaway. I would also like to thank the staff of the Department of Chemical and Biomolecular Engineering (CBE), namely Carmen Ghossein, Amber Tipton, and Sarah Large, for their assistance with travel, purchasing, and finances, and especially for the multiple late-night calls when flights were canceled or delayed.

I gratefully acknowledge the research collaborations made possible through the Center for Nanophase Materials Sciences (CNMS) at Oak Ridge National Laboratory, especially Dr. Michael Zachman, Dr. David Cullen, Dr. Linda Amichi, Dr. Haoran Yu, and Dr. Thomas Pfeifer. I extend my gratitude to the numerous members of the Breakthrough Electrolytes for Energy Storage Systems ($BEES^2$) Energy Frontier Research Center (EFRC) for their ongoing collaboration and support during my graduate studies. Finally, I would like to thank the United States. Department of Energy, whose funding made much of this work possible through the SMaRT program, the Bredesen Center, $BEES^2$ EFRC, and the GATE Fellowship.

ABSTRACT

Machine learning (ML) models are used to study chemical and materials systems. ML models correlate chemical properties to structural and feature-based representations. In addition, ML-based image analysis of scanning transmission electron microscopy (STEM) further illustrates how ML can be used to study these systems. Together, these projects enable high-throughput, data-driven analysis using ML models and open new possibilities for energy applications.

First, ML models predict key properties of chemical and materials systems. Graph neural networks (GNNs) and traditional ML models, including gradient-boosted trees and Gaussian processes, were evaluated for predicting these properties. The models predicted the equivalent alkane carbon number (EACN), which is associated with oil hydrophobicity and represents oil in the hydrophilic-lipophilic difference (HLD) equation, using a limited dataset, with performance evaluated across different geometry optimizations. The critical micelle concentration (CMC) is the concentration at which various electrochemical properties shift. Augmented GNNs combining graphs with non-graph features, namely temperature, demonstrated improved accuracy.

Designing μEs to maximize energy density requires the μEs to be liquid at room temperature without solvent dilution. Therefore, the melting point (MP) was constrained to best predict room-temperature liquid oils by using Coulomb matrices (CMs) that represent the intranuclear electrostatic interactions. When the oil is not liquid at room

temperature, solubility predictions minimize oil dilution and maximize energy density. Variational autoencoders (VAEs) were used as graph encoders to represent both the solute and the solvents separately, enabling multi-system predictions.

Then, STEM image analysis was conducted using ML. Generating a simulated STEM dataset with unbiased segmentation to train a mask R-CNN instance segmentation model capable of identifying, localizing, and outlining nanoparticles, even in cases of particle overlap where conventional image analysis methods fail. The mask R-CNN model successfully predicted per-instance segmentation masks for nanoparticles.

These projects contribute novel techniques and provide insight into how ML can be adapted to more effectively predict chemical properties and interpret microscopy data, expanding the scope of ML applications in chemical and energy research, and enabling high-throughput material and molecular discovery.

TABLE OF CONTENTS

Chapter 1: Introduction 1

Motivations 2

Redox Flow Batteries 5

Microemulsions as Electrolytes for Redox Flow Batteries 8

Porous Electrodes in Electrochemical Energy Devices 10

Research Objectives: Five Critical RFB-μE Properties 11

Machine Learning and Its Scientific Applications 24

Data Collection and Processing 34

Machine Learning Models 38

Chapter 2: Comparison of Machine Learning Approaches for Prediction of the Equivalent Alkane Carbon Number for Microemulsions Based on Molecular Properties 66

Abstract 67

Introduction 68

Methods 72

Results and Discussion 80

Conclusion 94

Chapter 3: Predicting Critical Micelle Concentration and the Effects of Temperature on Micellization using Machine Learning 97

Abstract 98

Introduction 98

Methods..... 102
Results and Discussion 116
Conclusion 137
Chapter 4: Coulomb Matrix Approach for Predicting Melting Point of Organic Molecules 139
Abstract 140
Introduction..... 141
Methods..... 151
Results and Discussion 153
Conclusion 164
Chapter 5: Graph Encoders for Predicting Non-Aqueous Molecular Solubility..... 166
Abstract 167
Introduction..... 167
Methods..... 170
Results and Discussion 190
Conclusion 212
Chapter 6: Machine Learning Approach for Detecting Nanoparticles in Fuel Cell Materials Via STEM Imaging..... 214
Abstract 215
Introduction..... 216
Methods..... 229

Results and Discussion 237

Conclusion 255

Chapter 7: Conclusion 256

References 262

Appendix 332

Vita 348

LIST OF TABLES

Table 2.1. Total run time in seconds of MPNN, CGCNN, and GCN with unoptimized vs. optimized geometries using each force field, MMFF94, and GAFF, and XGBoost on data generated using molecular descriptors predicting EACN. 81

Table 2.2. MAE, RMSE, and R^2 for the training and testing datasets for each model evaluated (MPNN, CGCNN, GCN, and XGBoost). 83

Table 3.1. Selected Hyperparameters for the three GNNs; MPNN, CGCNN, and GCN trained using data with geometries optimized using GAFF. 105

Table 3.2. Selected Hyperparameters for XGBoost trained using molecular descriptors. 107

Table 3.3. Selected Hyperparameters for the Gaussian Process trained using molecular descriptors. 109

Table 3.4. Training runtime in seconds of MPNN, CGCNN, and GCN trained using geometries optimized with GAFF. XGBoost, and GP on data generated using molecular descriptors predicting CMC. 117

Table 3.5. MAE, RMSE, and R2 for the training and testing datasets for each model evaluated (MPNN, CGCNN, GCN, XGBoost, and GP) trained using CMC data. Additionally, the MPNN, CGCNN, and GCN have been compared with different molecular geometries, graph data optimized using GAFF, and molecular descriptors used with XGBoost and GP. 120

Table 3.6. MAE, RMSE, R^2 for the training dataset for each model evaluated (MPNN, CGCNN, GCN, XGBoost, and GP) trained using the full CMC dataset. Evaluated on three groups: cationic, anionic, and nonionic. 132

Table 3.7. MAE, RMSE, R^2 for the testing dataset for each model evaluated (MPNN, CGCNN, GCN, XGBoost, and GP) trained using the full CMC dataset. Evaluated on three groups; cationic, anionic, and nonionic. 133

Table 4.1. Selected RDKit and PaDEL descriptors for MP predictions. 147

Table 4.2. CV testing results for each of the three models; LightGBM, XGBoost, and CatBoost, evaluated using MAE, RMSE, and R^2. 154

Table 4.3. Comparison of ML model results using MAE, RMSE, and R^2 using the testing set or equivalent to predict MP with the best results bolded. 163

Table 5.1. Dataset size (After removing incompatible molecules), total and unique molecule count, average molecule size, and average BertzCT of the five datasets (ZINC, QM9, Merged, BigSolDB, and Eval). 179

Table 5.2. Data descriptors used from RDKit and PaDEL. 185

Table 5.3. VAE testing results for the solute and solvent datasets with their BO hidden size, node loss, edge loss, KL loss, total loss, and graph similarity. 194

Table 5.4. VAE testing results for the solute and solvent datasets with their BO hidden size, node loss, edge loss, KL loss, total loss, and graph similarity, scaled to account for duplicate molecules within BigSolDB. The best testing results are bolded..... 200

Table 5.5. XGBoost training and testing results [-log(S)], in MAE, RMSE, and R^2..... 203

Table 5.6. Comparison of ML model results using MAE, RMSE, and R^2 using testing set or equivalent using log(S), with the best results bolded. 206

Table 5.7. Predicted vs True solubility values in [mol/L] and [-log(S)] for 21 solutions from table 2 in Ahn et al (Eval-solutes and Eval-solvents). The solution numbers use the same numbering system as the ROM column in table 2 of Ahn et al.[8] 208

Table 6.1. R-CNN testing metrics across all test images. 238

Table 6.2. R-CNN testing metrics across all test images separated by particle overlap. 239

Table 6.3. Distribution of particles per image, Feret diameter, area, and aspect ratio of true and detected particles. 243

Table 6.4. Distribution of particles per BOT image, Feret diameter, area, and aspect ratio of detected particles. 251

Table A.1. List and descriptions of RDKit descriptors 332

LIST OF FIGURES

Figure 1.1. Electrolytes of interest to BEES2 EFRC are represented along with bulk (H-bonding network dynamics, proton conduction, structure, etc.) and interfacial (electron transfer, redox reactions, etc.) properties of importance. (Used with permission from Dr. Burcu Gurkan, CWRU; BEES EFRC Director)..... 4

Figure 1.2. RFB schematic. Created in BioRender. Furth, N. (2026) https://BioRender.com/ykvrvn8..... 6

Figure 1.3. Overview of μEs using the Winsor system. Created in BioRender. Furth, N. (2026) https://BioRender.com/i4g1qc9..... 9

Figure 1.4. Overview of research objectives and impact. Created in BioRender. Furth, N. (2026) https://BioRender.com/037wgar 12

Figure 1.5. Journal articles from 2000-2025 using "Machine Learning & Physics", "Machine Learning & Chemistry", and "Machine Learning & Materials" as key words, sourced from Scopus. Accessed on January 22nd, 2026..... 30

Figure 1.6. Perceptron overview. Created in BioRender. Furth, N. (2026) https://BioRender.com/4tp33dg..... 39

Figure 1.7. Basic neural network example. Created in BioRender. Furth, N. (2026) https://BioRender.com/rjk5bix..... 40

Figure 1.8. Simple convolution operation with a stride of one and a filter size of 2x2.. 44

Figure 1.9. Simple max pooling operating with a stride of one and a filter size of 2x2.. 45

Figure 2.1. ML pipeline for predicting EACN. Created in BioRender. Furth, N. (2026) https://BioRender.com/t0xzjcl 74

Figure 2.2. RFECV plot for XGBoost, with the vertical dotted line highlights the optimal number of features.. 77

Figure 2.3. MAE, RMSE, and R^2 for the EACN model predictions. (A) Training MAE, (B) Training RMSE, (C), Training R^2, (D) Testing MAE, (E) Testing MRSE, and (F) Testing R^2. Nonoptimized geometries are shown with no hatches, GAFF optimized geometries are shown with dot hatches, and MMFF94 optimized geometries are shown with circle hatches. CGCNN is shown in blue, GCN is shown in orange, MPNN is shown in green, and XGBoost is shown in red..... 84

Figure 2.4. GNN model loss curves with training error in blue and validation error in orange..... 87

Figure 2.5. Regression plots for each model tested with training prediction shown in black, testing predictions shown in solid red. Ideal fit is shown as a solid blue line and model fit shown in a red line. 95% confidence intervals shown in shaded blue, and the 95% prediction interval shown as gray dashed lines..... 88

Figure 2.6. Distribution of Errors for the three GNNs with EACN data with different geometry optimization, (A-I) and (J) molecular descriptors with XGBoost with the training set in black and the testing set in red. 90

Figure 2.7. Testing MAE and RMSE for each category (Alkane, Alkene, Alkyne, Aromatic, Cyclic, Cyclic Alkane, Cyclic Alkene, Cyclic Ester, Ester, Ether,

Halogenated Alkane, Ketone, Nitrile, Substituted Aromatic, Terpene, Terpenoid, sand Triglyceride), for the three GNNs with EACN data (A-I) and (J) XGBoost. With MAE shown in solid bars and RMSE shown in hashed bars. Categories which do not appear in the testing set are shown with no bar. The overall MAE is shown as a black solid horizontal bar and the overall RMSE is shown as a black solid horizontal bar. Functional groups with ten or less molecules in the dataset are shown with an * in the legend. 92

Figure 3.1. ML workflow for GNNs (Top) and XGBoost/Gaussian process (Bottom). Molecular structure and corresponding SMILES representation (inputs); data processing and model training for GNNs consists of SMILES to XYZ plots with geometry optimization followed by MatDeepLearn which generate adjacency matrices to atom and bond feature vectors with an additional feature vector for GNN model training. Data processing and model training for XGBoost and Gaussian Process consists of SMILES to molecular descriptors, followed by feature elimination and model training both GNNs and XGBoost/Gaussian Process output regression plots and evaluation metrics; MAE, RMSE, and R^2. Created in BioRender. Furth, N. (2026) https://BioRender.com/ayu9zgd 103

Figure 3.2. Histogram of A) CMC in molarity (M), B) CMC in -Log(M), and C) temperature distribution in Celsius (°C). 111

Figure 3.3. Illustration of appending a temperature surrogate node to graph data. 113

Figure 3.4. Recursive feature elimination with cross-validation graph for CMC. The vertical dotted line highlights the optimal number of features used in the XGBoost and GP models. 115

Figure 3.5. Training (black) and testing (red) results ((A) MAE, (B) RMSE, and (C) R^2) for the three GNNs with CMC data, CGCNN, GCN, and MPNN trained with data optimized using GAFF and molecular descriptors with XGBoost Gaussian Process. 119

Figure 3.6. Regression plots for the three GNNs with CMC data, (A) CGCNN optimized using GAFF, (B) GCN optimized using GAFF, (C) MPNN, optimized using GAFF (D) molecular descriptors with XGBoost, and (E) molecular descriptors with Gaussian Process with the fit line of the model shown in black, the 95% confidence interval shown in blue, the 95% prediction limit shown as a dashed grey line, and the ideal fit as a blue line. Training data is shown in black, and testing data is shown in red. Horizontal bands are shown with a black arrow. 125

Figure 3.7. Distribution of absolute errors for each model evaluated (CGCNN, GCN, MPNN, XGBoost, and Gaussian Process) scaled to the same axis. (A) CGCNN optimized using GAFF, (B) GCN optimized using GAFF, (C) MPNN optimized using GAFF (D) molecular descriptors with XGBoost, and (E) molecular descriptors with Gaussian Process, with the training set shown in black and the testing set shown in red. 127

Figure 3.8. Error versus Temperature plots for each model evaluated (CGCNN, GCN, MPNN, XGBoost, and Gaussian Process). (A) CGCNN optimized using GAFF, (B) GCN optimized using GAFF, (C) MPNN, optimized using GAFF (D) molecular descriptors with XGBoost, and (E) molecular descriptors with Gaussian Process. 129

Figure 3.9. Distribution of headgroup classes: Cationic data (Blue), Anionic data (Orange), and Nonionic data (Green). 131

Figure 3.10. Training (black) and testing (red) results for the three GNNs with CMC data, CGCNN, GCN, and MPNN trained with data optimized using GAFF and molecular descriptors with XGBoost Gaussian Process with each category (cationic, anionic, and nonionic). (A) MAE for cationic, (B) RMSE for cationic, (C) R^2 for cationic, (D) MAE for anionic, (E) RMSE for anionic, (F) R^2 for anionic, (G) MAE for nonionic, (H) RMSE for nonionic, and (I) R^2 for nonionic. 134

Figure 4.1. Unprocessed Coulomb matrix examples with their structures for A) Phenol, B) Catechol, and C) Ferrocene. 145

Figure 4.2. Data collection and generation workflow, including duplicate removal, Coulomb matrix generation and processing, RDKit and PaDEL descriptor generation, and Morgan fingerprint generation. 148

Figure 4.3. SHAP values (red-blue gradient) of the top 12 features, with CMs and ECFPs aggregated. 150

Figure 4.4. Regression plots for MP predictions for A) LightGBM, B) XGBoost, and C) CatBoost. CV test points are shown as black dots, Benik et al. evaluation points as red

dots, the fit line of each model as a red line, the ideal fit line as a blue line, the 95% prediction interval as gray dashed lines, and the 95% confidence interval as shaded blue..... 157

Figure 4.5. Error distributions for A) LightGBM, B) XGBoost, and C) CatBoost shown as density as a function of error in ºC. 159

Figure 4.6. Training and Validation loss curves for A) LightGBM, B) XGBoost, and C) CatBoost..... 160

Figure 5.1. Histograms for A) solubility [mol/L], B) log-solubility [-log(S)], and C) temperature after outlier removal..... 177

Figure 5.2. A) glycerol, B) 2,2,6,6-tetramethyl-1-piperidinyloxy (TEMPO) C) methane, created using ChemDraw. 181

Figure 5.3. ML workflow to predict solubility using VAEs and BO (top) to generate an implicit representation (latent vectors) of the solutes, solvents, and temperature (middle). Then, using the generated representations, train an XGBoost model to predict solubility (bottom). 186

Figure 5.4. SHAP values (red-blue gradient) of all model features with each set of latent values and molecular fingerprints aggregated. 188

Figure 5.5. Bayesian optimization results for all three models A) the ZINC-VAE, B) the QM9-VAE, and C) the Merged-VAE, with GP means (Blue), 95% confidence intervals (Red), five initial points (Blue), five subsequent points (Orange), and the optimal point (Yellow)..... 192

Figure 5.6. Principal component analysis (PCA) for the latent vectors of the three models trained A) (ZINC-VAE), B) (QM9-VAE), and C) (Merged-VAE). Labels for ZINC (Blue), QM9 (Orange), BigSolDB-solutes (Green), BigSolDB-solvents (Red), Eval-solutes (Purple), and Eval-solvents (Brown). 196

Figure 5.7. Principal component analysis color coded by solvent species. 201

Figure 5.8. A) Regression plots for solubility predictions with the CV testing sets (Black), testing set (Red), the true prediction line (Red), the fit line (Blue), and the 95% prediction interval (Grey), B) distribution of standard deviation of errors (residuals), and C) distribution of predictions with a standard deviation greater than three as a function of the true solubility [-log(S)]..... 204

Figure 5.9. Loss curves five-fold cross group-wise testing with XGBoost. 210

Figure 6.1. Example of generated data (A) HAADF-STEM image generated using PySlice, (B-D) particle masks generated for each particle in the HAADF-STEM image.... 226

Figure 6.2. Poisson (shot) noise samples along the entire range in SNR. 228

Figure 6.3. Example of processed data (A) HAADF-STEM image generated using PySlice with Gaussian noise, Gaussian blur, Laplacian sharpening, CLAHE, and Poisson noise applied (B-D) particle masks generated for each particle in the HAADF-STEM image..... 230

Figure 6.4. BOT image sample from Yu et al..... 231

Figure 6.5. Tiling example of the BOT data with the first three tiles shown. 232

Figure 6.6. Examples of processed BOT images with Laplacian sharpening and CLAHE. 233

Figure 6.7. ML workflow from PySlice data generation (Orange) to Mask R-CNN training (Purple) to HAADF-STEM imaging (Red) to real data evaluation (Green). 236

Figure 6.8. Mask R-CNN metrics over all overlap levels. Precision (Blue), recall (Orange), F1-score (Green), accuracy (Red), mean IoU (Purple), and median IoU (Brown). 240

Figure 6.9. True (Blue) versus predicted (Orange) distributions for particles per image and particle overlap..... 244

Figure 6.10. True (Blue) versus predicted (Orange) distributions for particle Feret diameter, area, and aspect ratio. 245

Figure 6.11. Sample of particle detection with the PySlice test data. 248

Figure 6.12. Predicted distributions for BOT particle count per images, Feret diameter, area, and aspect ratio. 252

Figure 6.13. Sample of particle detection with the BOT data..... 253

CHAPTER 1:

INTRODUCTION

Motivations

Machine learning (ML) accelerates discovery in chemical and materials systems for energy storage by predicting electrolyte properties and analyzing electrode microstructures where traditional methods fall short. Experimental measurements provide high accuracy but low throughput, whereas quantitative structure-property relationship (QSPR) models offer interpretability at moderate accuracy but generalize poorly across diverse chemistries. Computational approaches such as density functional theory (DFT) deliver mechanistic insight but scale poorly for high-throughput screening of vast chemical spaces. The central challenge is not the absence of data alone, but the need to rapidly prioritize chemical candidates and extract quantitative morphological information from experiments that are too slow or labor-intensive to perform exhaustively.

The work in this dissertation applies ML to address key bottlenecks in redox flow battery (RFB) electrolytes, focusing on chemical properties relevant to μE design and image analysis for nanoparticle detection in porous electrodes via scanning transmission electron microscopy (STEM) imaging. Such ML-driven capabilities directly support next-generation energy storage research, where innovative electrolyte formulations promise enhanced power density and cycling stability.

Advancing these capabilities requires innovative electrolytes to support grid-scale integration, prompting the U.S. Department of Energy (DOE) Office of Basic Energy Sciences (BES) to fund the Breakthrough Electrolytes for Energy Storage (BEES) Energy

Frontier Research Center (EFRC) in 2018 (through DE-SC0019409), renewed as Breakthrough Electrolytes for Energy Storage and Systems (BEES^2) in 2022.[1,2] BEES^2 is a collaboration between eight universities and one national lab (Case Western Reserve University (Lead), The University of Tennessee-Knoxville, Texas A&M University, New York University, The University of Notre Dame du Lac, Hunter College of the City University of New York (CUNY), The University of Illinois, Ohio State University, and Pacific Northwest National Laboratory), a summary the scope of BEES is shown in **Figure 1.1, including** CoHBEs and μEs.

From 2018 to 2022, BEES pioneered the study of novel electrolytes structured at the molecular to mesoscopic scale, building a fundamental understanding of their physical, transport, and electrochemical properties through collaborative experimental and theoretical efforts with systems such as μEs and soft nanoparticle electrolytes. Building directly on this foundation, BEES^2 studies the transport mechanisms of ions, protons, redox-active species, and electrons in structured electrolytes across bulk and interfacial environments. The development of ML-based property prediction and image analysis workflows supports this mission by reducing trial-and-error in electrolyte formulation and improving throughput in electrode characterization. These systems are studied both in bulk and at electrode-electrolyte interfaces to enable high energy and power density in next-

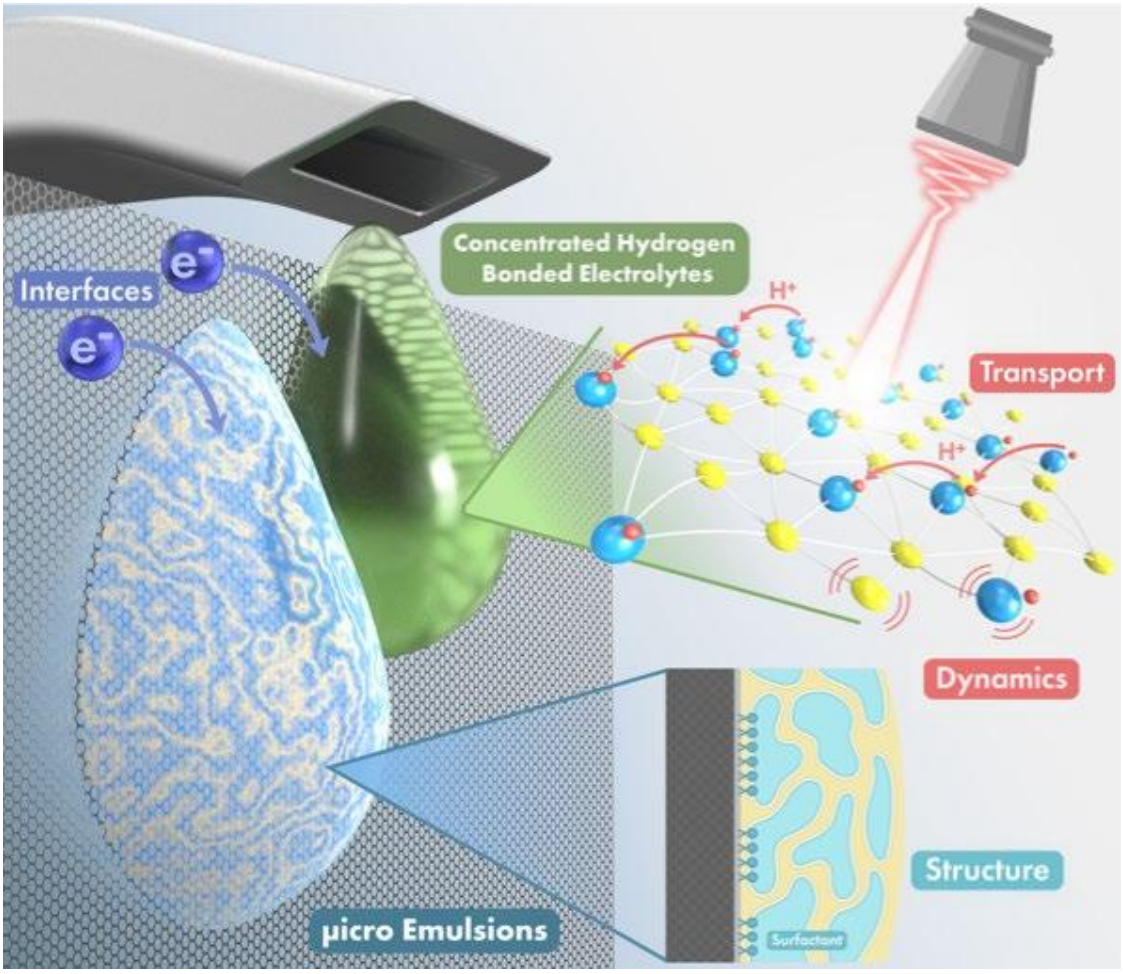

Figure 1.1. Electrolytes of interest to BEES2 EFRC are represented along with bulk (H-bonding network dynamics, proton conduction, structure, etc.) and interfacial (electron transfer, redox reactions, etc.) properties of importance. (Used with permission from Dr. Burcu Gurkan, CWRU; BEES EFRC Director).

generation energy storage systems such as RFBs. Building on BEES studies of structured electrolytes and μEs, BEES^2 advances design principles to decouple energy density from conductivity, focusing on protic concentrated hydrogen-bonded electrolytes (CoHBEs) for proton-coupled electron transfer (PCET) and μEs to study proton transport, redox-species transport, and interfacial electron transfer, through two thrusts focused on bulk dynamics and transport, and interfacial electron transfer and stability. This evolution supports decarbonization by integrating renewables and has broader impacts in electrocatalysis and separations.[1,2]

Redox Flow Batteries

RFBs are scalable energy storage devices that use liquid electrolytes stored in external reservoirs. These electrolytes flow into a cell stack with porous carbon electrodes (PCEs) separated by an ion-exchange membrane, which prevents electrolyte mixing (membrane crossover) while allowing ion transfer to drive redox reactions during charging and discharging.[3–8] The work presented in this dissertation applies ML to accelerate discovery in chemical and materials systems for energy storage, focusing on two areas: (1) predicting μE properties as alternative RFB electrolytes, and (2) analyzing STEM images for nanoparticle characterization in the PCE, which is also useful for proton exchange membrane fuel cells (PEMFCs).

As shown in **Figure 1.2,** RFBs use two reservoirs: one for the anolyte and one for the catholyte. In an RFB, the electrolytes are pumped through porous electrodes and

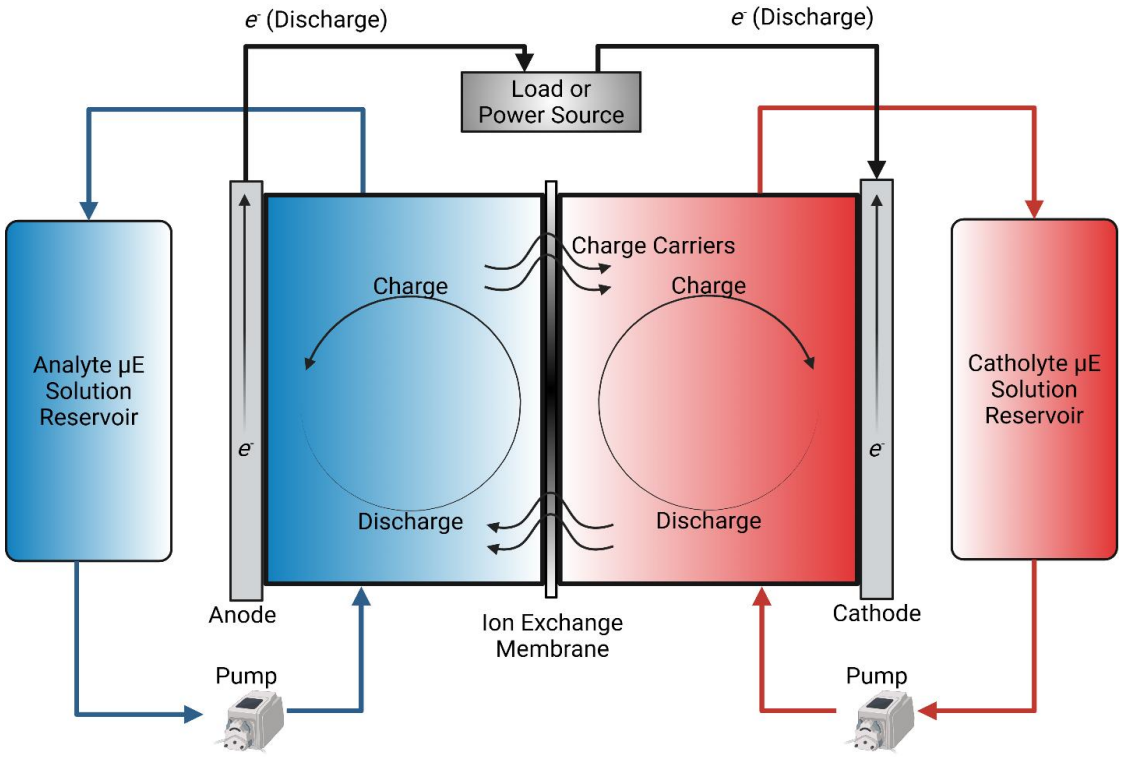

Figure 1.2. RFB schematic. Created in BioRender. Furth, N. (2026) https://BioRender.com/ykvrvn8

separated by an ion-exchange membrane. Drawing an external current drives ion transport across the membrane, coupling oxidation and reduction reactions at the respective electrodes. Together, these redox reactions enable the RFB to charge and discharge. RFBs offer key advantages, including: (1) scalability, where energy capacity increases by enlarging the reservoir volume, (2) longevity, demonstrated by over 10,000 charge-discharge cycles and 10+ year lifespans with minimal degradation; and (3) stability, achieved through decoupled power and energy components that eliminate overcharge needs and enable low self-discharge during idle periods.[9–12].[7,13,14] Despite these advantages, RFBs face challenges, including low power density; vanadium-based RFB systems, which dominate commercial applications, typically achieve only 25-35 Wh/kg, compared to lithium-ion batteries at ~250 Wh/kg.[15,16] Membrane crossover, in which ions diffuse through perfluorosulfonic acid (PFSA) membranes such as Nafion, leading to self-discharge and capacity fade; and membrane degradation, accelerated by acidic environments that cause chemical breakdown and reduced ion selectivity over time.[11,17–19] To overcome these drawbacks, significant research is underway to discover new electrolytes for RFB applications, particularly within the BEES2 EFRC, where μEs solubilize redox-active species in aqueous surfactant systems. These systems achieve higher power density than vanadium-based systems, though μEs introduce stability challenges, including phase and interfacial instabilities.[2,5]

Microemulsions as Electrolytes for Redox Flow Batteries

A μE is a thermodynamically stable, isotropic, homogeneous mixture of oil, water, and surfactant, frequently incorporating a co-surfactant (typically a short-chain alcohol) that lowers the interfacial tension required for spontaneous nanoscale domain formation.[5,6,20–43] In RFB applications, the oil phase can solubilize redox-active organic species; the aqueous phase provides ionic conductivity. The surfactants stabilize the oil-water interface, lower interfacial tension, and enable the formation of nanoscale domains.[20,21,27,44] In 1948, Winsor classified μEs into four equilibrium types based on phase behavior and curvature, which follows work from Hoar and Schulman in 1943.[45] The Winsor system is shown in **Figure 1.3**.[46] The four Winsor types are as follows:

- Type I (oil-in-water, O/W): Oil-swollen micelles with surfactant-stabilized oil nanodroplets dispersed in excess water.
- Type II (water-in-oil, W/O): Inverse micelles encapsulating water droplets in continuous oil, with excess aqueous phase.
- Type III (bicontinuous with excess): Interpenetrating oil and water domains (middle-phase μE) coexisting with excess oil/water layers.
- Type IV (homogeneous): Complete bicontinuous across all component proportions.

The BEES^2 EFRC targets type IV μEs to directly address the low energy density of vanadium RFBs while maintaining thermodynamic stability through optimal oil-surfactant

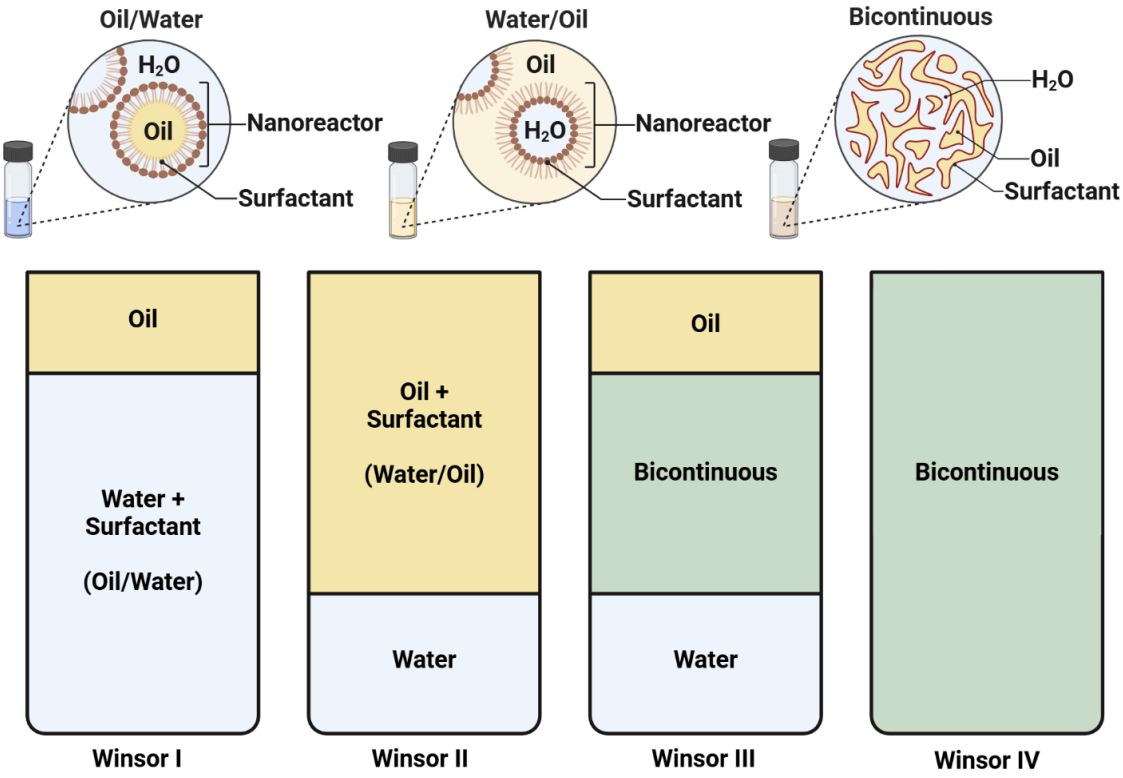

Figure 1.3. Overview of μEs using the Winsor system. Created in BioRender. Furth, N. (2026) https://BioRender.com/i4g1qc9

selection.[5,27] Winsor type can be estimated using the hydrophilic-lipophilic difference (HLD) equation shown in equations 1.1 (ionic) and 1.2 (nonionic).

$$HLD_{Ionic} = \ln(S) - K \times EACN - f(A) - \alpha_t \Delta T + Cc \quad (1.1)$$

$$HLD_{Nonionic} = b \times S - K \times EACN - \Phi(A) - c_t \Delta T + Cc \quad (1.2)$$

In equations 1.1 and 1.2, ln (S) and $b \times S$ represent the salinity of the μE, $K \times EACN$ represents the oil phase, $f(A)$ and $\Phi(A)$ represent the co-surfactant, $\alpha_t \Delta T$ and $c_t \Delta T$ represents the temperature, and Cc represents the surfactant.[21,23,29,31,47–50] HLD ≈ 0 corresponds to types III/IV (ideal bicontinuous/continuous phases for RFB electrolytes); negative HLD yields type I; positive HLD yields type II. As RFB anolyte/catholyte, μEs interface directly with the CE to facilitate reversible redox, ion storage, and high charge-transfer rates.

Porous Electrodes in Electrochemical Energy Devices

Commonly used in RFBs, PCEs offer affordability, corrosion resistance, and a high surface area for electrochemical reactions.[14] Nanoparticles can be incorporated into these electrodes to improve energy efficiency, charge transfer, and reduce ohmic losses. However, poorly designed nanoparticle coatings can compromise stability and block pores

within the electrode. Quantifying nanoparticle loading, size distribution, and overlapping from STEM images is therefore important for understanding how electrode structure affects performance. Electrolyte design and electrode structure are coupled problems, because the practical performance of an RFB depends not only on the molecular properties of the electrolyte but also on how effectively the electrode microstructure supports wetting, transport, and accessible surface area. Another application of PCEs are as PEMFC catalyst layers, where particle size, overlap, and distribution strongly influence catalytic activity and durability.[13,51] Quantitative particle statistics provide a measurable bridge from microscopy to electrode performance, allowing microstructural differences to be compared across materials, processing conditions, and aging states. Because overlapping particles are difficult to resolve with conventional geodesic active contour and watershed segmentation methods, ML-based image analysis provides a more reliable route for high-throughput characterization.[51] This data-driven strategy extends from molecular property prediction to mesoscale characterization, where automated particle detection helps quantify microstructure in energy materials.

Research Objectives: Five Critical RFB-μE Properties

The work in this dissertation studies five aspects of RFBs: the EACN of oils, the CMC of surfactants, the MP of oils and solubility of organic molecules across solvents, and the distribution and detection of nanoparticles within electrode materials at the electrode–electrolyte (μE) interface, an overview is shown in **Figure 1.4**. Each of these properties has its own challenges to measure experimentally, making ML a valuable tool to overcome

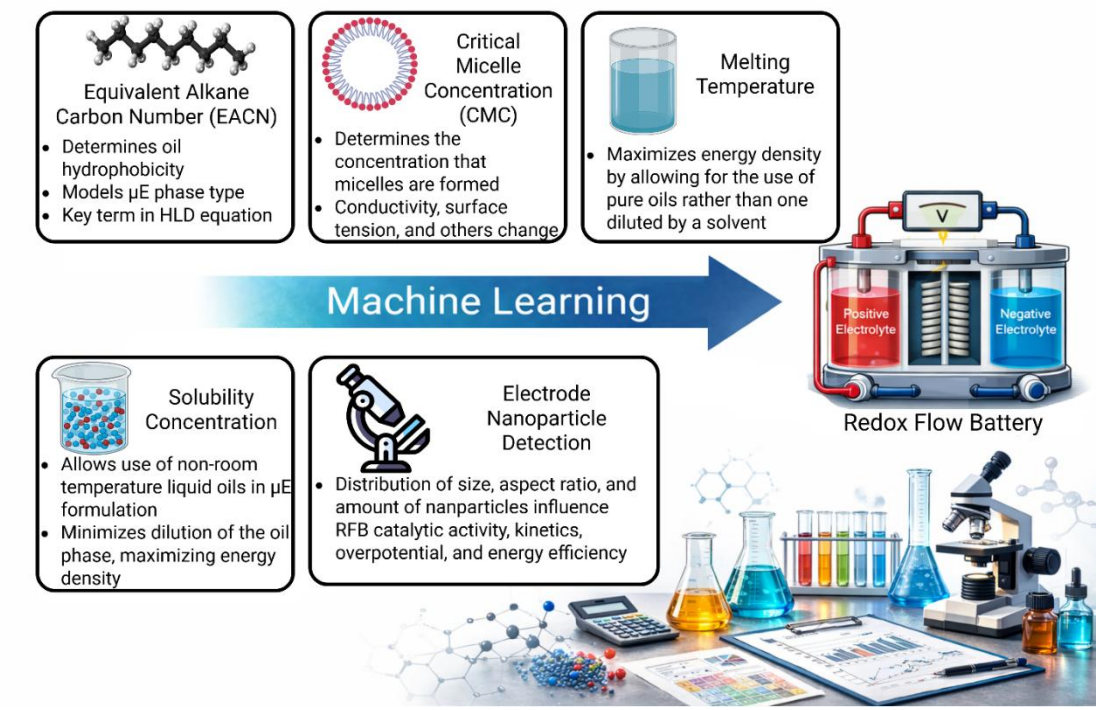

Figure 1.4. Overview of research objectives and impact. Created in BioRender. Furth, N. (2026) https://BioRender.com/037wgar

these obstacles. Taken together, studying each of these five components of RFBs allows for high-throughput screening of different components to optimize RFB power density, electrolyte stability, and commercial viability.

- The EACN influences μE phase behavior in the type III/IV region through its role in the HLD equation and also affects oil hydrophobicity, oil loading, and interfacial tension.[52,53]
- The CMC influences surfactant aggregation and related interfacial properties such as surface tension, viscosity, osmotic pressure, and conductivity.[54–59]
- The MP influences whether the oil phase remains liquid under operating conditions, which affects phase behavior, μE formation, and RFB cycle life.[60–62]
- The solubility of the oil influences the concentration range over which type III/IV μEs can form, which in turn affects oil loading, phase separation behavior, surfactant efficiency, and electrolyte homogeneity.[60,62,63]
- Nanoparticle distribution in PCEs influences active site accessibility, charge transfer resistance, ohmic losses, pore blockage risk, cycling stability, and electrode lifespan.[64–66]

Together, these studies establish a practical ML workflow for energy materials research that couples property prediction, high-throughput screening, and automated image analysis to support accelerated experimental decision-making.

The EACN represents the equivalent behavior of a molecule as a pure alkane carbon chain and represents the molecule as such; however, unlike the number of carbons in a pure alkane chain, the EACN can be both fractional and negative. Additionally, the EACN can estimate hydrophobicity and is a key term within the HLD equation, as shown in equations 1.1 and 1.2. [22,52,67–73] The HLD equation models phase behavior for ionic and nonionic μE systems, and the EACN represents the oil term in the HLD equation. Experimentally, measuring the EACN requires very pure oils and surfactants (> 98%), with a surfactant that equalizes the hydrophobicity of the oil to prevent dissolution.[70,73,74] Generally, Low-EACN oils often favor type I μEs. In contrast, higher-EACN oils can more readily access the type III/IV region, although phase type is determined by the full formulation rather than the oil alone.[52,53,66]

To bypass experimental limitations, several computational approaches have been developed to predict EACN directly from molecular structure. Lukowicz et al. applied the conductor-like screening model for real solvents (COSMO-RS), in which σ-moments (σ_0, σ_2) of the surface charge density are used as descriptors to predict EACN.[70] Their model correlates well with experimental EACN for several functional groups: esters, ketones, and alcohols, well capturing polarity-induced reductions relative to nominal carbon number and providing mechanistic insight into how surface interactions control μE hydrophobicity. Building on the work of Lukowicz et al., Delforce et al. introduced two fast models: a graph machine (GM) that learns directly from SMILES strings and a neural network (NN) that uses COSMO-RS σ-moments as inputs.[52] The GM is optimized via cross-validation (CV)

on complex cosmetic oils; both achieve similar accuracy, with the GM excelling on homologous series trends and the σ-moment NN exploiting quantum-derived features.

Complementary empirical work by Acree et al relates the EACN to Abraham solute parameters using only five transferable descriptors, yielding strong alignment with fish-tail method data, which determines EACN by observing the characteristic 'fish-tail' curvature in μE phase diagrams where optimal phase behavior occurs. This alignment shows that Abraham solute parameters can offer a route for high-throughput EACN screening.[68,75] At a more macroscopic level, Yoga et al. used the hydrophilic-lipophilic difference net average curvature (HLD-NAC) framework, which uses the HLD equation as input to measure net and average curvature to predict EACN for various oils by simultaneously fitting solubility data over dilution and pressure. By assigning methane an effective carbon number of 1.0 and incorporating pressure corrections, resolving inconsistencies in traditional interpolation schemes, and producing physically consistent EACN values that improve microemulsion formulation in EOR.[76] These works show that EACN can be predicted from descriptors with both computational-based methods, such as COSMO-RS and ML. Still, existing models predict narrow ranges of oils, trained on small datasets, and aren't tailored to μE electrolytes. In contrast, the work in this dissertation (1) further benchmarks graph-based and descriptor-based models on a broader EACN dataset from different sources, (2) analyzes errors by functional group, demonstrating how different representations and models differ in accuracy across functional groups, and (3) focuses on the HLD application for type III/IV μE electrolytes in RFBs, turning EACN prediction into

a practical tool for μE electrolyte design and HLD parameter rather than a standalone property fit.

The CMC is a critical property of the surfactant representing the concentration at which micellization begins.[43,77–90] A micelle is a spherical aggregate of surfactant molecules with the hydrophobic tails at the oil interface and hydrophilic heads at the water interface.[78–88] At the CMC, when micelles first form, several key properties of the solution change: conductivity,[89,91] surface tension,[43,77,89] osmotic pressure,[89] detergency,[43] and interfacial tension.[88] Several of these are relevant to BEES^2 research with RFBs; micellization and surfactant aggregation influence ion mobility and interfacial properties in the μE electrolytes, affecting energy and power density, interfacial tension effects type III/IV μE formation, and the lower surface tension at and above CMC improves electrode wetting, resulting in higher surface area for redox reactions.[54–59] CMC can be determined using various techniques, including NMR, surface tension measurements, and computational approaches such as COSMO-RS.[77,80,88,90,92] Each technique requires specialized equipment or software, which can be expensive, require training, and is limited by throughput.

Existing CMC prediction methods span graph models, descriptor-based ML, hybrid ML- QSPR, and physics-based simulations. Brozos et al. showed that GNNs trained on molecular graphs can learn CMC alongside other surfactant properties across diverse headgroups and tails, using transfer and ensemble learning to improve robustness across measurement techniques. They demonstrated that graph representations capture key

structural drivers of micellization (tail length, branching, headgroup identity) without manual descriptor engineering.[93] Chen et al. instead used RDKit descriptors with several ML models: extreme gradient boosting (XGBoost), random forest (RF), graph convolutional network (GCN), Gaussian process (GP), and multiple linear regression (MLR), reducing 208 features to 19 via recursive feature elimination with cross-validation (RFECV); Shapely additive explanations (SHAP) analysis revealed temperature dominates nonionic CMC, modulated by hydrophobicity, headgroup size, and branching.[85] Seddon et al. combined ML with the Szyszkowski equation to fit surface-tension isotherms and derive and validate CMCs for hydrocarbon surfactants, surfacing non-obvious structural descriptors.[92,94] Earlier, Huibers et al. showed that within homologous nonionic families, a minimal three-descriptor linear model (chain length, headgroup hydrophilicity, branching) accurately predicts CMC when the chemistry is narrow.[86] Nivón-Ramírez et al. used COSMO-RS-parametrized dissipative particle dynamics (DPD) simulations to reproduce the sodium dodecyl sulfate (SDS) CMC mechanistically.[88] More recent work from Gonzalez-Nunez et al and Nnadili et al. has expanded the modeling space further, with transformer-based molecular language models trained using SMILES, outperforming GNNs over thousands of surfactants by capturing longer-range amphiphile motifs, and hybrid pipelines using atomistic simulations plus GPs to model temperature and salinity-dependent CMC surfactants.[95,96]

Together, these studies show that CMC is predictable from structure and conditions using both graph-based and descriptor-based ML, hybrid ML-QSPR, and physics-driven

simulations. Still, most efforts focus on single surfactant classes or idealized aqueous systems. Still, most efforts focus on single surfactant classes or idealized aqueous systems, and they rarely target μE electrolytes for energy storage or examine temperature effects across the operating range relevant to RFBs. The work present in this dissertation builds upon these foundations by (1) developing CMC models that explicitly couple structural features with temperature to support screening of surfactants for type III/IV μEs in RFB applications, (2) benchmarking graph-based models against interpretable descriptor-based models on mixed surfactant chemistries, and (3) demonstrating their utility for high-throughput surfactant screening across headgroups (cationic, anionic, nonionic) and broad temperatures (0-90°C).

For RFB applications, determining which redox-active oils are liquid at room temperature enables μEs to form under standard operating conditions.[97] Additionally, oils that are liquid at room temperature allow for the use of pure oil, rather than oils that need to be diluted by a solvent, maximizing the oil fraction and by extension the power of the RFB. For μE electrolytes in RFBs, the MP also affects type III/IV μE formation range because solid oil crystals can disrupt surfactant packing, making it difficult to form a type III/IV structure, resulting in the formation of a type I μE. Finally, the MP of the redox-active oils affects the lifespan of the RFB, as frequent melting and solidification during thermal cycles (RFB operation, day-night, and seasons) can clog both the CE and electrolyte pumps, potentially causing the RFB to fail prematurely.[60–62] While experimentally determining the MP of a single species is straightforward, the synthesis and

testing of each candidate redox-active oil is prohibitively slow. As a result, determining the MP of redox-active oils in a high-throughput manner is not experimentally feasible, making computational methods and ML desirable.

Properties such as MP, glass transition temperature, and solubility (logS) govern phase behavior across materials science and energetics, though prediction remains challenging due to crystal packing, lattice energy, and polymorphism.[98,99] Recent work spans SMILES strings, descriptors, GNNs, and natural language processors (NLPs), often finding that tree-based and graph-convolutional-based models outperform more complex ML methods. However, solid-state effects remain hard to encode. Tao et al. used the PolyInfo dataset with SMILES-derived atomic, morphological, and descriptor features across several models, with RF providing the best results, and no correlation between model complexity/structure and prediction quality.[98] In Mi et al., the NLP model treated canonicalized SMILES as word sequences, using data augmentation to improve generalization and capture longer-range packing patterns.[100] Galeazzo et al. used an NN to predict MP for 200k compounds using functionality embeddings that distinguish isomer groups.[99] Polyakov et al. trained RF, XGBoost, and an NN model on Cambridge Crystallographic Data Center (CCDC) MP data using RDKit descriptors, and estimated logS via the general solubility equation.[101,102] In St John et al., GNNs with simple atom- and bond-level features outperformed ECFP baselines for MP and solubility. Hughes et al. benchmarked QSPR for logS, MP, and logP, finding MP the hardest due to descriptor gaps

in crystal packing.[103] Even advanced embeddings and GNNs plateau at ~26–30°C MAE, suggesting an intrinsic limit with current representations and modeling techniques.

Existing MP models provide useful screening but remain error-limited due to poor encoding of intermolecular crystal-packing effects. In contrast, the work presented in this dissertation (1) uses compressed representations of CMs to explicitly capture pairwise intranuclear electrostatic interactions relevant to melting, unlike descriptors or graph encodings that do not encode this physics as directly, (2) compiled the largest MP dataset to date, and (3) targets room temperature liquid prediction to enable undiluted type III/IV μE electrolytes, providing practical high-throughput guidance for energy storage.

When the redox-active oil is not liquid at room temperature, it needs to be solubilized by dilution, thereby reducing the power density of the μE and, consequently, the RFB. Unlike the EACN, the CMC, and the MP, solubility involves multiple components to consider: solute, solvent, and temperature. The Type III/IV μE formation range is affected by solubility, requiring a higher surfactant fraction and an oil concentration because higher solubility results in greater homogeneity in the μE; thus, power density is affected due to the dilution of the solvent.[60,62,63] Measuring the solubility of a single solution reliably requires careful control of various factors, including temperature, solution agitation, and equilibrium time. Because of the number of components and the complexity of the process, measuring the solubility of a variety of solutions across a range of temperatures in a high-throughput manner is time-consuming

and impractical, making ML desirable. However, to broadly predict a complex property such as solubility, a large-scale database is needed.

Krasnov et al. developed the BigSolDB dataset, containing 103,944 experimental solubility values across different solutes, solvents, and temperatures, providing the scale needed for ML to predict such a complex property accurately.[104] Perryman et al. used just 14 compact descriptors with multiple ML models to predict solubility in both organic and aqueous systems, demonstrating that minimal feature sets can rival large descriptor batteries. Shimizu et al. trained SVMs to approximate expensive DFT calculations for argon gas solubility across solvents, enabling fast predictions from molecular inputs alone. Qiao et al. pruned the AstraZeneca and MLSMR datasets (59,510 compounds total) to train Bayesian classifiers for aqueous solubility (>100 μmol/L threshold), demonstrating that activity-based pruning improves generalization and training speed. Krishnan et al. used AEs to explore chemical space via latent perturbations, in which similar molecules cluster in low-dimensional representations. Coupry et al. developed graph-based similarity metrics combining isomorphism, kernels, and ML for robust molecular comparisons beyond single benchmarks.

Existing approaches target aqueous logS or gas solubility but rarely address non-aqueous oil-solvent solutions used in μEs. This work (1) uses separate graph encoders (VAEs) for solute and solvent to model multi-component non-aqueous solubility, (2) benchmarks against state-of-the-art descriptor-based baselines on RFB-relevant systems,

and (3) integrates predictions into HLD workflows to minimize dilution while maintaining type III/IV stability.

Nanoparticles in PCEs affect catalytic activity and reaction kinetics in RFBs by increasing the number of active sites and the surface area.[13,14,51,105] Accurately detecting and analyzing nanoparticle quantity and size distribution is vital. Yet, manual particle detection methods fail at scale due to the nanoparticles' nm-scale dimensions relative to electrode dimensions. Human inspection is impossible without electron microscopy methods such as STEM imaging. However, single STEM images span the μm scale, requiring thousands of images for an entire electrode. Additionally, manual particle delineation introduces human bias, particularly when particles overlap.[51] PySlice was used to generate synthetic STEM images with known ground-truth masks, controlled noise, and adjustable overl, enabling the segmentation model to be trained and evaluated on data where manual labeling would be prohibitively slow and biased. Analyzing nanoparticle distributions allows assessment of electrode wetting effectiveness, thereby inferring the probability of pore blockage, cycling stability, and electrode lifespan.

Zhang et al. reviewed ML applications for electron microscopy (EM) imaging, spanning transmission electron microscopy (TEM), scanning electron microscopy (SEM), and STEM defect recognition, phase classification, hyperspectral signal extraction, and in-situ time-series analysis of video frames, demonstrating the versatility of ML across grayscale, spectral, and temporal data modalities.[106] Alldritt et al. trained NNs on simulated/experimental atomic force microscopy (AFM) images to infer 3D atomic

structures and bond motifs from tip-sample interactions, outperforming feature-engineered methods on noisy data.[107] Kim et al. developed an ML workflow for SEM particle sizing/morphology, using angular bisecting lines, automated thresholding, and watershed segmentation to isolate overlapping core-shell nanoparticles and extract geometric parameters at high-throughput. Li et al. tiled EM micrographs into windows for CNN-based defect detection (voids, dislocations, precipitates), achieving orders-of-magnitude speedup over manual inspection with improved localization accuracy.[108] Yu et al applied automated HAADF-STEM + energy dispersive X-ray spectroscopy (EDS) across proton exchange membranes fuel cell cathodes, analyzing >100k Pt nanoparticles pre/post-cycling via geodesic active contours and watershed segmentation; spatially resolved size, density, and composition revealed inhomogeneous degradation (Pt loss near membrane, bimodal Co distributions), validated against small angle X-ray scattering. [51]

Existing methods excel at segmentation but struggle with overlapping STEM particles in porous CE supports under RFB conditions. The work in this dissertation (1) trains an R-CNN model using simulated STEM datasets with unbiased ground-truth overlap segmentation to enable nanoparticle detection, (2) benchmarks against geodesic/watershed baselines for size/distribution accuracy, and (3) targets RFB electrode optimization by quantifying pore blockage and wetting from distributions, supporting high-throughput electrode design.

Studying these five aspects of RFBs using ML overcomes the challenges of experimental and classical high-throughput computational methods, offering a trade-off

between speed and accuracy. Despite this trade-off, the ML models serve as valuable high-throughput screening tools for predicting potential oil and surfactant candidates for μEs, and for automated analysis of electrode nanoparticles using STEM imaging. Taken together, this work demonstrates how ML can reduce the cost, time, and uncertainty associated with electrolyte design and electrode analysis, advancing BEES^2 goals for structured, high-performance energy storage materials.

Machine Learning and Its Scientific Applications

Before discussing ML, first, the methods that preceded it are reviewed: experimental measurements, QSPR models, and computational chemistry. Each method has inherent trade-offs among accuracy, speed/throughput, interpretability, and generalizability.

Experimental measurements offer the highest accuracy through direct determination and the highest interpretability via observable physical mechanisms, with universal generalizability to any measurable system. Efforts to measure and correlate chemical properties with structure date back to the 19th century, when chemists used empirical observations of boiling points, densities, and reactivities across homologous series to identify trends and relationships among related compounds.[109,110] By the early 20th century, systematic experimentation ranging from calorimetry and tensiometry for thermodynamic properties to spectroscopy for electronic structure became the cornerstone of physical chemistry, enabling precise characterization of molecular behavior in isolation

or solution.[109] These methods provide unmatched high accuracy and interoperability, but their dependence on pure samples and sensitivity to conditions such as temperature and impurities limit throughput to dozens or hundreds of compounds, making them ideal for validation yet insufficient for screening vast chemical spaces in applications like electrolyte design.[111] Finally, given their role as the foundation of QSPR, computational methods, and ML, experimentation is highly generalizable across all problems and molecules/materials.

QSPR modeling provides moderate accuracy and high interpretability. QSPR emerged in the mid-20th century as an extension of quantitative structure-activity relationships (QSAR), with early examples in the 1960s-1970s correlating simple topological indices or constitutional descriptors with properties such as boiling points in alkanes or vapor pressures in organic compounds.[112,113] Pioneered by workers such as Corwin Hansch and Toshio Fujita, QSPR formalized linear and nonlinear regressions between hand-crafted molecular descriptors, such as lipophilicity (logP), polar surface area, and connectivity indices, and measured properties, thereby achieving interpretable predictions within narrow chemical families.[112] While computationally cheap and insightful for extrapolation within domains, reliance on expert-selected features in QSPR often leads to poor generalization across a wide range of chemistries or uncharted functional groups, as seen in limitations for predicting surfactant behaviors or nanoparticle distributions. QSPR provides moderate accuracy and high interpretability through transparent descriptor-property maps, with high-throughput, near-instant predictions post-training.[103,110,114,115]

Quantum-mechanical methods revolutionized property prediction, beginning in the 1920s-1930s with the Schrödinger equation.[116,117] Although practical computational chemistry took off in the 1960s-1970s via semi-empirical approaches and ab initio calculations, it evolved into density functional theory (DFT) after the 1965 Hohenberg-Kohn theorems and the 1964 Kohn-Sham formalism.[116,118–120] Since the 1990s, density functional theory (DFT) has become a widely used quantum chemical method for computing electronic structure, geometries, energies, and related properties such as vibrational spectra and, with appropriate models, solvation effects. Extensions such as COSMO-RS can be used to estimate solvent-dependent thermodynamic properties, and molecular dynamics is often used to capture finite-temperature motion.[118,121,122] These physics-based tools offer mechanistic depth and transferability without experiments, thereby enabling predictions of reaction pathways and phase behavior. Computational methods offer high accuracy and interpretability, but their high computational cost and poor scalability limit them to focused studies rather than high-throughput virtual screening for energy materials.[103,118,121]

In this setting, ML is most valuable when experiments are costly, the search space is large, and reliable experimental or computational screening or measurement would otherwise require impractical time and resources.

While each preceding method dominates specific dimensions, ML uniquely balances high accuracy, unprecedented throughput, and excellent generalizability across the vast chemical spaces critical to RFB electrolyte design, albeit with interpretability

challenges that modern those techniques address. ML models have found increasingly extensive use in scientific applications, including in energy applications such as those studied in chemistry, physics, and materials science, as shown in **Figure 1.5**. In BEES^2, Dr. Tuckerman at NYU and Dr. Maginn at Notre Dame use ML to accelerate molecule and material discovery through automated pipelines that integrate large language models for dataset generation, ML interatomic potentials for atomistic simulations, and sigma profile generation.[123–126] The work in this dissertation contributes to BEES^2 by focusing on high-throughput prediction of μE properties and the detection of nanoparticles at the electrode-electrolyte interface, with greater emphasis on applications than on fundamentals.

Fundamentally, ML models operate as intricate optimization functions that search for parameter values that minimize an error function in a high-dimensional space, thereby providing an optimal mapping of model inputs to generate accurate outputs. The first known use of the term "Machine Learning" was by an IBM employee, Arthur Samuel, in 1959; the methods that laid the foundation of ML were developed and studied millennia earlier.[127] The earliest use of mathematical optimization to find the minima of a function used the critical points theorem by utilizing the derivative. The derivative is a mathematical concept described by Sir Issac Newton and Gottfried Wilhelm Leibniz in the late 17th century, measuring the instantaneous rate of change of a function.[128] Newton and Leibniz incorporated the work of the ancient Greek mathematician Archimedes who studied tangents in the 3rd century BC.[129] Derivatives can be expanded into high-dimension space as a Jacobian matrix, mapping each variable as a partial derivative of every other variable

developed by Carl Gustav Jacob Jacobi in 1841.[130,131] Derivatives can be further adapted for high-dimensional functions by utilizing the Hessian matrix, which used second-order derivatives developed in the 19th century by Ludwig Otto Hesse.[132] The expansion of derivatives to the Jacobian and Hessian matrices enabled ML optimization methods such as backpropagation and gradient descent. Gradient descent is a crucial method used in various ML models, including NNs and gradient boosting trees (GTBs). Gradient descent is an optimization method that utilizes partial derivatives to find the minima of a multivariable function. Gradient descent was first suggested in 1847 by Augustin-Louis Cauchy, with the convergence properties for non-linear applications described by Haskell Curry in 1944.[133,134] While these early methods on their own are insufficient to solve the complex optimization problems that ML models are now used for, they laid the foundation for many modern ML methods, including the NN and other gradient-based models such as GTBs. Tree-based and descriptor-based models provide strong baselines for small datasets; in contrast, graph-based models capture structure more directly, and image analysis models extend the framework to latent representations and microscopy data.

In the 1960s, complex models such as the multi-layered perceptron (MLP) were developed based on the perceptron unit that was first described in 1955 by Frank Rosenblatt.[135] However, the perceptron performs poorly using data with no strong linear correlations. To overcome this limitation, the MLP was first described by Alexy Grigorevich Ivakhnenko in 1965, using sets of perceptrons to make predictions on non-linear data, thereby overcoming a key limitation of the basic perceptron unit.[136–139] More

complex NNs, such as convolutional neural networks (CNNs) and RNNs, were developed later in the 1970s and 1980s.[140–142] Modern NNs use the backpropagation algorithm, shown in equations 1.29 and 1.30, which applies the Leibniz chain rule.[143] The creation and implementation of the backpropagation algorithm was considered as a significant impact because it updates the parameters of the model layer-wise as opposed to standard gradient descent, which updates all parameters in a single step.[127] The invention of backpropagation allowed for a resurgence of the research and use of ML after ML became less utilized during the preceding short "ML Dark Age" or "AI Winter", despite the development of key methods used in modern ML, such as the first autoencoder (AE)-like model.[144,145] However, ML development and utilization would again slow in the late 1980s and early 1990s due to the limited capability of ML models at the time.[106,146–157]

After this short ML dark age, advancements in ML developed rapidly. Several key developments and achievements, including the loss of chess grandmaster Gary to Deep Blue in 1997, showed much promise.[158] The development of the SVM and the GP in 2006 provided high-performing models without the need for gradient descent or partial derivative calculations, unlike NNs.[159–161] In 2008, the development of the GNN, a specialized model that can operate on graph data, has proven particularly useful for molecular graphs.[13,22,52,162–169] Another important advancement is the variational auto-

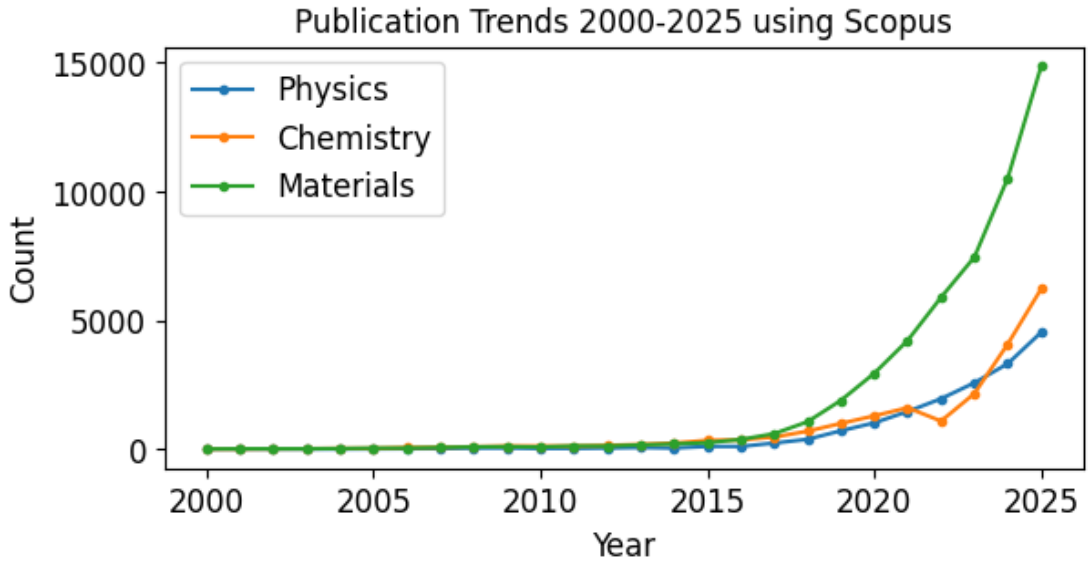

Figure 1.5. Journal articles from 2000-2025 using "Machine Learning & Physics", "Machine Learning & Chemistry", and "Machine Learning & Materials" as key words, sourced from Scopus. Accessed on January 22nd, 2026.

encoder (VAE) developed by Ian Goodfellow in 2014; a variant of the standard AE that learns distributions over the latent space instead of fixed values, which has been key to the development of generative AI, generative adversarial networks (GANs), and molecular graph generation.[166,170,171] Advancements in computer vision, such as the U-Net[172] and ImageNet[173] architectures, also opened key applications such as live video object detection, which is a powerful tool for microscopy image analysis,[51,174,175] energy storage monitoring, control, and optimization,[176,] and commercial applications such as self-driving cars,[177] Face-ID,[178] and traffic management.[141] In 2015, GPs and NNs were combined to create "Deep Kernel Learning," in which the final layer of an NN is a GP model that approximates a dense layer with an infinite number of neurons, thereby enabling the uncertainty measurements provided by GPs.[179] This uncertainty quantification is especially important in these settings because it helps distinguish high-confidence predictions from regions where the model is likely extrapolating beyond the training data. Around the same time, the now-popular and powerful GTB model, the extreme gradient boosting (XGBoost) algorithm, was developed.[180]

More recently, large language models (LLMs) such as ChatGPT, Copilot, and Perplexity, based on the GPT-4 and later GPT-5 LLMs, have become extremely popular both for leisure and for practical applications.[181,182] These more complex models became feasible due to the readily available computing power of "supercomputers" such as Frontier and Summit at ORNL and ISAAC at the University of Tennessee, as well as more powerful personal computers and workstations on a smaller scale.[183–186] Feasibility also greatly

benefited from codebases becoming open-source rather than proprietary or otherwise unpublished, and from the introduction of open-source programming libraries such as SciKit-Learn, TensorFlow, Keras, and PyTorch.[187–189]

While ML models demonstrate substantial promise across various scientific domains, significant limitations persist: achieving acceptable predictive accuracy is exceedingly challenging, or deploying an ML model is otherwise impractical. ML excels when clear correlations exist between inputs and targets, whether linear or nonlinear, with a plethora of models to choose from, each with its own advantages and disadvantages. In contrast, absent or obscured correlations often prevent reliable predictions, rendering feature engineering a critical prerequisite for success. Exemplifying this, seemingly straightforward properties such as the MP of organic substances are readily measured by differential scanning calorimetry yet remain difficult for ML because they depend strongly on crystal packing and electrostatic interactions, whereas commonly used molecular descriptors often do not encode these effects directly, as described by Hughes et al.[190] Datapoints exhibiting similar input features but with disparate targets pose difficulties, particularly for spatially-based models such as k-NN and SVMs, which assume smoother decision boundaries.[191] A major limiting factor in ML is the amount of available data for real-world applicable problems and the quality of that data, which is especially important when applying ML to scientific applications. Insufficient dataset sizes preclude the extraction of high-level representations essential for generalization beyond the training-set distribution, often resulting in poor ML models prone to overfitting. Easily automated

problems in an experimental setting or problems that can already be conducted in a high-throughput manner would not greatly benefit from ML, as real-world experiments will yield near-exact values rather than predictions. Problems well known to be solvable using QSPR would not benefit from ML. Likewise, domains where transducing inputs into machine-readable formats proves arduous or infeasible, e.g., wavey bonds that prevent SMILES generation due to unspecified stereochemistry or a mixture of isomers, or non-finite representations of data, such as some crystals, which often have an infinite repetition, fall outside the purview of ML.

As ML continues to advance, its integration with traditional scientific applications is transforming how discoveries are made. Beyond serving as predictive or optimization tools, ML models are now enabling fundamental insights into physical and chemical mechanisms that were previously hidden within complex, multidimensional data.[192,193] The fusion of ML with experimental and theoretical research promotes a feedback loop in which models not only accelerate hypothesis testing but also guide experimental design, enabling more targeted and efficient exploration of materials and energy systems.[194] In this way, ML is not replacing scientific reasoning but rather augmenting it, empowering scientists to uncover relationships, patterns, and governing principles that bridge computation, experimentation, and theory, driving the next generation of breakthroughs in chemistry, physics, and materials science.[195]

Data Collection and Processing

Large molecular databases, such as; the national institute of standards and technology (NIST),[196] PubChem,[197] BigSolDB,[104] QM9,[198] ZINC,[199] open quantum materials database (OQMD),[200] computational materials repository (CMR),[201] inorganic crystal structure database (ICSD),[202] crystallography open database (COD),[203] ChemSpider,[204] Cambridge structural database (CSD) which maintains the CCDC dataset,[205] online chemical modeling environment (OChem),[206] organic materials database (OMDB),[207] and the American mineralogist crystal structure database (AMCSD)[208] each contain thousands to millions of molecules with known molecular characteristics, structures, and properties. When a large database is not already available, data can be collected from other sources, such as individual journal articles and books, generated using Python simulations, such as the Atomic Simulation Environment (ASE), or ab initio transmission electron microscopy (abTEM), or modeled using QSPR.[209–211] Calculations from modeling programs, such as DFT and COSMO-RS software, can also be used to generate data.[212] However, the format and representation of the raw data are often unstructured, inconsistent between databases, or otherwise inadequate for ML training without processing the data into a consistent, machine-readable format.

To make raw data useful for ML training, each molecule must first be transformed into a machine-readable representation, such as a SMILES string. If not already included in the database, this is typically done by querying PubChem or the NIH using the molecule's name to find a known SMILES.[197,213] For molecules without a known SMILES

in existing databases, a molecular drawing program such as ChemDraw can be used to draw the molecule and generate the SMILES from the drawn structure.[214] The SMILES can then be converted into a set of molecular descriptors, typically using packages such as RDKit, DScribe, and PaDEL.[215–217] Alternatively, depending on the problem and the user's domain knowledge, descriptors can be calculated using known equations, such as QSPR, to generate values that may better fit the problem than generic molecular descriptors. Another type of representation offered by RDKit is extended connectivity molecular fingerprints (ECFPs), which are bit vectors representing a set of substructures within a molecule, as shown in detail by Rogers et al.[218] RDKit generates the ECFPs by evaluating atom and bond types, as well as the substructures present in the molecule. In addition to their use as descriptors, ECFPs are often used to calculate molecular similarity, with Tanimoto similarity being the most common measure.[218,219] However, other methods exist for defining molecular similarity, as well as alternative ways to specify the elements of the bit vector that constitute the ECFPs. No single representation is optimal for all properties; using multiple encodings helps match the representation to the target property and requires sufficient domain knowledge.

In addition to traditional descriptors, SMILES can be used to generate a molecular graph that contains implicit structural features rather than explicit molecular descriptors. Graph representations are useful when traditional descriptors do not capture aspects of the target property; they often require more data to capture lower-level features due to the complexity of GNNs.[52,93,162,165–169,219–230] The molecular structures are represented as

molecular graphs, initially composed of adjacency matrices and node and edge feature vectors, which together provide a holistic view of the molecule in a form compatible and useful for GNNs. The adjacency matrices are binary $n \times n$ matrices, where n is the number of heavy atoms (non-hydrogen atoms), that describe which atom pairs are connected. Node feature vectors encode atomic attributes such as atomic number and charge, while edge feature vectors encode bond type and related connectivity information. To generate molecular graphs, SMILES are first converted to a three-dimensional XYZ representation, either as an .xyz or .json file, and then, using a package such as ASE, the molecular graphs are generated for use in a GNN model.

Additionally, the molecular geometry can be optimized using a force field. The general amber force field (GAFF) and the Merck molecular force field 94 (MMFF94) are the most common in the literature. They can be used to optimize molecular structures with respect to bond angles, bond lengths, and torsion angles.[231,232] Both GAFF and MMFF94 are designed for organic molecules, although other force fields exist that are optimized for different types of molecules. Geometry optimization boosts performance over raw structures but increases computational cost and training time. Generally, the force field applied doesn't directly affect the adjacency matrix or atomic indices themselves, although it provides more accurate atomic distances and positions used by various frameworks.

Another representation that can be derived from the SMILES is CMs.[216] A CM represents molecules using their intermolecular nuclei, first introduced in Rupp et al., to maintain representation and energy during molecular rotation.[216,233] A CM with N atoms can be

generated by taking the elements along the diagonal, basing these values on the atomic numbers of the atoms, while other indexes not on the diagonal in the CM define the attraction between two atoms based on their three-dimensional distance. The equation for calculating CM values is shown in equation 1.3.

$$M_{ij}^{Coulomb} = \begin{cases} 0.5Z_i^{2.4} for\ i = j \\ \frac{Z_i Z_j}{R_{ij}}\ for\ i \neq j \end{cases} \quad (1.3)$$

In equation 1.3, i and j are atom indexes, R_{ij} is the distance between atoms i and j, Z_i is the atomic number of atom i. The diagonal elements are interactions within each atom, and the off-diagonal elements are the Coulomb repulsions between nuclei i and j. CMs do not change information based on the orientation of a molecule, because they are symmetric with a shape of $N \times N$. Since the molecules in most datasets are of vastly different sizes, and most ML models require all inputs to be of heterogeneous dimensionality, the matrices need to be converted to a standard size, either by resizing or by using the largest eigenvalues, with the number of eigenvalues determined by the smallest. Since CMs are symmetric across the diagonal, the values above the diagonal are duplicates of those below; as a result, they can be removed to save memory and storage space without losing any information.

Machine Learning Models

The fundamental unit of an NN is the perceptron shown in **Figure 1.6**, originally used for binary classification. The perceptron takes a set of input features X_n, multiplies them by corresponding weights w_n then sums the resulting products and adds an optional constant bias term b to the summation. The final value then passes through an activation function; common choices include sigmoid, tanh, or the rectified linear unit (ReLU) for regression tasks, and a threshold function for classification, producing the model's prediction. During training, the model updates its weights using gradient descent until it reaches a set number of iterations or after the per-iteration improvement falls below a set threshold, enabling early stopping. However, a single perceptron can only make accurate predictions for linear data.

NNs expand the capabilities of the basic perceptron to handle non-linearly separable data with high dimensionality. A NN, shown in **Figure 1.7**, is an ML model comprised of layers containing multiple perceptrons (neurons) connected with weighted edges. A NN consists of three types of layers: an input layer, hidden layers, and an output layer. The input layer receives the data and has a dimensionality equal to that of the data. After the input layer, each hidden layer has a set of nodes, often with an activation function, connected by weighted edges. Hidden layers can include multiple types: dense, pooling, convolutional, and dropout layers. In a standard NN, typically only dense and dropout layers are used. The output layer then provides the model's prediction, returning either a

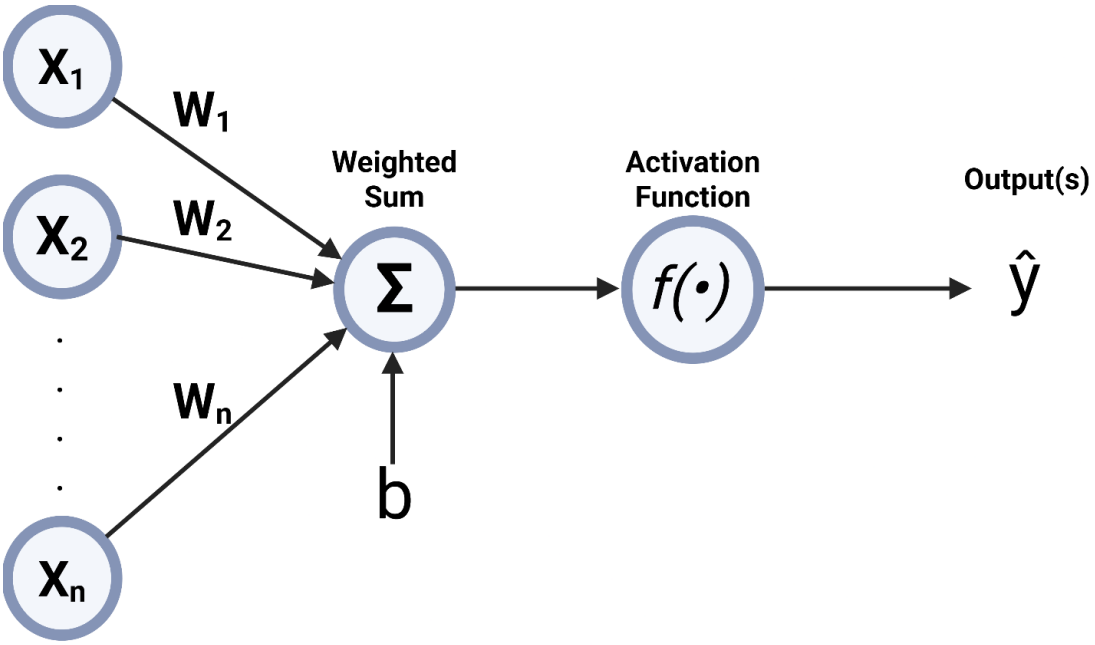

Figure 1.6. Perceptron overview. Created in BioRender. Furth, N. (2026) https://BioRender.com/4tp33dg

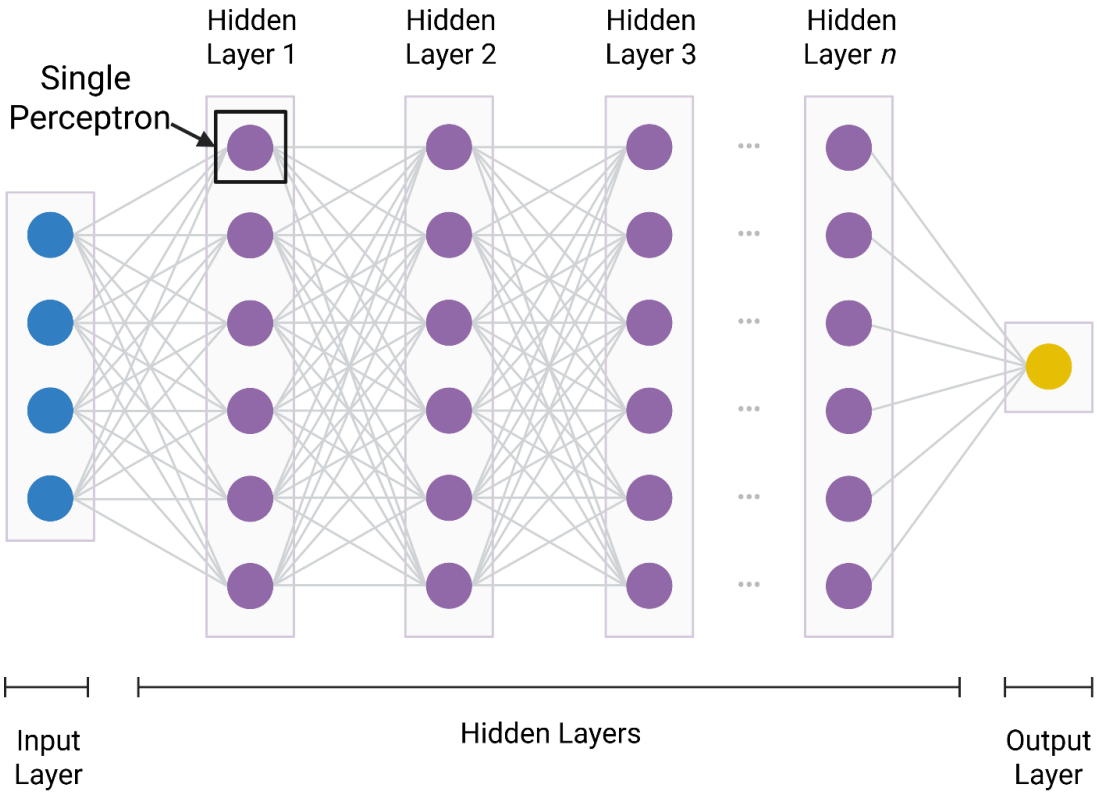

Figure 1.7. Basic neural network example. Created in BioRender. Furth, N. (2026) https://BioRender.com/rjk5bix

single continuous value for regression tasks or a probability distribution for classification tasks.

During training, an optimizer function, such as stochastic gradient descent (SGD) and its variants, is used. SGD is used to find the optimal set of gradients that minimizes the loss function by making small changes to the parameters model of the model θ_n.

Given data x and targets y, (x, y) is the training set, and α is the learning rate of the model, a scalar that controls the size of each SGD step. Typically, the learning rate will start at a high value to get close to the global optimum, while avoiding becoming trapped in a local minimum, and then decrease to fine-tune the model parameters. Finally, a loss function is used to evaluate the model's performance during training by comparing its predictions with the target value. The loss function the models optimize varies depending on the task; common losses include mean absolute error (MAE), used for regression tasks (equation 1.24), and cross-entropy (CE) and variants thereof, used for classification tasks (equations 1.25 and 1.26). Regularization terms can vary, but typically take two forms: L1 regularization, which penalizes all model parameters equally, and L2, which penalizes larger parameters more due to the square term. As a result of adding regularization terms,

$$\mathcal{L}(\theta, x, y) = \underbrace{\sum_{i}^{n} |y_i - \hat{y}_i|}_{MAE\ Loss} + \underbrace{\lambda_1 \sum_{j} \theta_j}_{L1\ Regularization} + \underbrace{\lambda_2 \sum_{j} \theta_j^2}_{L2\ Regularization} \tag{1.4}$$

the model, in theory, optimizes its parameters for the distribution of the entire problem rather than the distribution represented by the training set, thereby avoiding overfitting to lower-level features. A complete loss function $\mathcal{L}(\theta, x, y)$ using MAE as an example, with both L1 and L2 regularization terms, can be expressed in equation 1.4, where x is the input data, y_i is the actual value, $\hat{y}_i$ is the predicted value, n is the number of data points, λ is the regularization strength, which is a scalar that is different for L1 and L2 regularization, and θ_j are the parameters of the model. Activation layers act like dense layers; however, unlike standard dense layers, they apply an activation function to each node's output. Common activation functions shown in equations 1.5-1.8 include Sigmoid, Softmax, ReLU, and Tanh, along with their variants.

Convolutional layers were introduced to handle high-dimensional data more efficiently, which differentiates CNNs from standard NNs. Convolutional layers use filters

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{1.5}$$

$$\sigma(x)_i = \frac{e^{x_i}}{\sum_{j=1}^{K} e^{x_j}} \; for \; i = 1, \ldots, K \; and \; x = (x_i, \ldots, x_K) \; \epsilon \; R^K \tag{1.6}$$

$$f(x) = \; x^+ = \frac{x + |x|}{2} \tag{1.7}$$

$$tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{1.8}$$

that multiply the input values by the filter's scalar weights and repeat this process until the entire input has been traversed. The scalar values in the filter are learned values, similar to parameters in NNs. The filter's scalars are trained in the same manner using SGD and backpropagation. The advantage of using a convolutional layer instead of a dense layer is that it reduces the number of operations while maintaining predictive accuracy. The backpropagation algorithm utilizes SGD; instead of optimizing the model parameters as a whole, each layer is optimized individually, leading to more optimal gradients.[234] The convolutional layer takes two arguments, the filter size, typically $n \times n$, and the stride, which controls the way in which the filter traverses the datapoint, typically $m \times m$. As shown in **Figure 1.8**, the learned convolutional filter starts at the first index of the data point, multiplies the values stored within it by the scalars in the filter, and then sums the products to generate a new value for the next layer. This process is repeated until the filter has traversed the entire datapoint.[173,235] Pooling layers typically follow convolutional layers, and take an input of higher dimensionality, and pool it to a lower dimensionality.[236] The most common pooling operations are min, max, and average pooling, which use the minimum, maximum, and average values within the filter, respectively, to produce a single output value. Shown in **Figure 1.9** is a simple pooling operation using the max pooling, the movement of the filter is the same as with the convolutional operation however, instead of using a sum of products to determine the value, the max value within the filter is used, for average pooling the average of the values within the filter would be used, both convolutional and pooling layers have variants designed for graph-based data in GNNs.

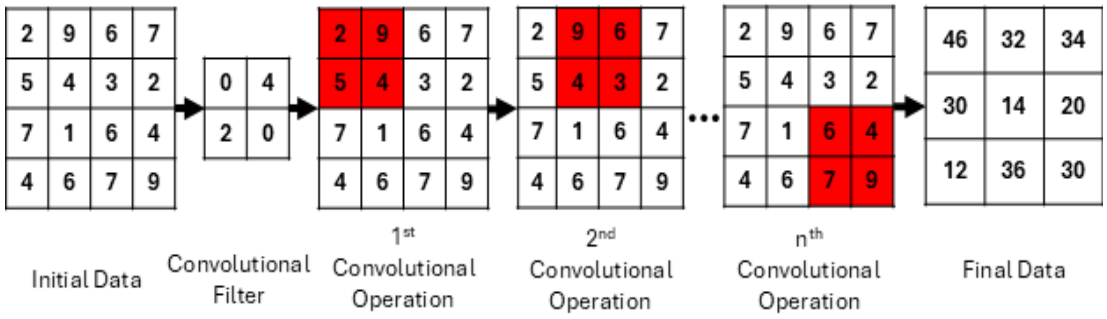

Figure 1.8. Simple convolution operation with a stride of one and a filter size of 2x2.

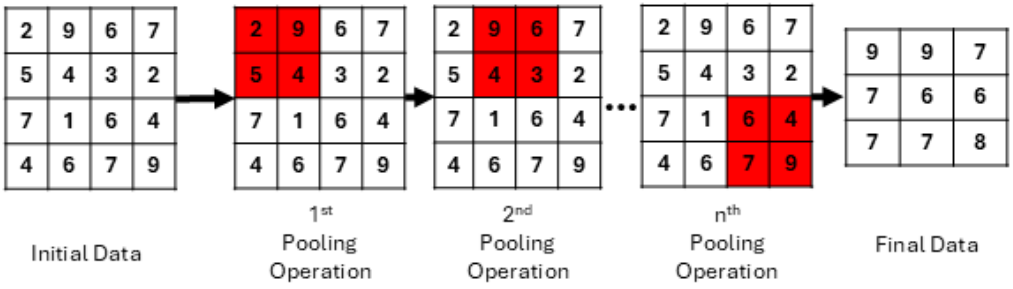

Figure 1.9. Simple max pooling operating with a stride of one and a filter size of 2x2.

Dropout layers permanently remove a fraction of the neurons in a layer with a fixed probability, aiming to prevent overfitting; this contrasts with gradient pruning, where a fixed fraction of edges with the smallest-magnitude parameters is dropped.[237] Typically, the output of the pseudo-random selected probability ρ neurons will be set to zero, with the output of the remaining neurons scaled by $\frac{1}{1-\rho}$ ensuring that the sum is the same. Dropout layers are especially useful when little data is available, as they ensure the model learns only the most high-level features of the data and serve as a form of regularization. In addition to preventing overfitting, dropout layers can speed up model training compared to non-pruned models dense layers due to needing fewer matrix and partial derivative operations during backpropagation.

A common representation of molecular data is the molecular graph, which has heterogeneous dimensionality and, as a result, cannot be used with standard NNs. To solve this problem, GNNs were introduced. GNNs are specialized NNs designed to operate on graph-based data as input.[22,52,93,166–169,222–230] Graph data, which has a non-linear structure with heterogeneous dimensions, can be used instead of the typical tabular and image data of fixed dimensionality. To address the heterogeneous nature of graph data, GNNs use specialized convolution layers at the beginning of the network, employing graph convolution operations that aggregate node information from neighbors, similar to how standard convolution operates. In addition to convolutional layers, GNNs use specialized pooling layers that iteratively reduce the graph to a fixed dimensionality while retaining relevant information. There are several types of GNNs commonly used in the literature to

solve problems in chemical systems, including the message-passing neural network (MPNN),[168,238] the crystal graph convolutional neural network (CGCNN),[168,169,228,230] and the GCN.[226] These GNNs, among others, can be easily implemented using existing frameworks such as MatDeepLearn, as demonstrated in Fung et al, an environment used to benchmark materials data with several state-of-the-art GNNs.[224]

The MPNN is a type of NN that utilizes a dense NN along with message-passing and update functions.[168,238] MPNNs utilize graph data, which contains a graph G, with vertex features x_i for vertex i, and edge features e_{ij} for an edge between vertex i and vertex j. To process the graph data, MPNN operates in two phases: the message-passing phase and the readout phase. The message passing phase tunes for T time steps and utilizes a message passing function, M_t, and a vertex update function, U_t. The message-passing functions pass data between neighboring vertices in the network by aggregating messages from each neighbor. The message passing function utilizes hidden states h_i^t at each node in the graph, and are updated using m_i^{t+1} and can be described as follows in equations 1.9 and 1.10,

$$m_i^{t+1} = \sum_{w \epsilon N(i)} M_t\left(h_i^t, h_j^t, e_{ij}\right) \tag{1.9}$$

$$h_i^{t+1} = U_t(h_i^t, m_i^{t+1}) \tag{1.10}$$

where $N(i)$ is the neighbor of the vertex i in graph G. The update vertex function U_t averages the previous state and the received message to create a gated recurrent NN. The exact function used for the vertex update function varies between applications and works. The readout phase computes a feature vector of the entire graph $\hat{x}$ using a readout function R defined as:

$$\hat{x} = R\left(\left\{h_i^T \middle| v \in G\right\}\right) \tag{1.11}$$

where $\hat{x}$ is used to produce the final predictions. The message-passing, vertex-update, and readout functions are all learned as differentiable functions.

The CGCNN model was developed for crystal structures and is adaptable to various types of molecules, such as crystals, polymers, and organic molecules.[168,169,228,230] The model uses a specialized convolutional operation to update the representations of vertices in the graph. The procedure computes the weighted sum of inputs and applies a gating function to generate updated vertex representations. The convolutional operation function used by CGCNN can be defined as in equations 1.12 and 1.13:

$$v_i^{(t+1)} = v_i^{(t)} + \Sigma_{j,k}\sigma\left(z_{(i,j)k}^{(t)}W_1^{(t)} + b_1^{(t)}\right) \odot g\left(z_{(i,j)k}^{(t)}W_2^{(t)} + b_2^{(t)}\right), (i,j)k \in G \tag{1.12}$$

$$z_{(i,j)k}^{(t)} = v_i^{(t)} \oplus v_j^{(t)} \oplus u_{(i,j)k}^{(t)} \tag{1.13}$$

where, $v_i^{(t)}$ is the embedding vector of a vertex in the graph, $W^{(t)}$ and $b^{(t)}$ represents the weights and bias matrices at the tth convolutional step, $\sigma(\cdot)$ is a sigmoid activation function shown in equation 1.5, $g(\cdot)$ is a nonlinear activation function, and $u_{(i,j)k}^{(t)}$ is a vector containing the cosine distances between neighboring vertices i and j as indexed by k. After completing T convolutions, the network learns the feature vector $v_i^{(T)}$ for each atom. A normalized summation is used as the pooling function, generating a feature vector v_c for the graph as follows:

$$v_c = Pool(v_0^{(0)}, v_1^{(0)}, \dots, v_N^{(0)}, \dots, v_N^{(R)}) \tag{1.14}$$

In addition to the convolutional and pooling layers, two fully connected (dense) layers with depths of L_1 and L_2 are added to capture the mapping between the molecular structure and the target property. The CGCNN model is effective at capturing local

structural features and has been successfully applied in fields such as chemistry and materials science.[165,169,228,230]

Lastly, the graph convolutional network (GCN) uses a spectral graph convolutional operation that transforms the graph into a spectral domain, enabling the model to capture structural information.[222,226] The convolutional network can be represented using a layer-wise propagation rule as follows in equation 1.15,

$$H^{l+1} = \sigma(\widetilde{D}^{-\frac{1}{2}}\tilde{A}\widetilde{D}^{-\frac{1}{2}}H^{l}W^{l}) \tag{1.15}$$

where $\tilde{A} = A + I_N$ is the sum of the adjacency matrix of the graph A_N and the identity matrix. $\widetilde{D}_{ii} = \Sigma_j \tilde{A}_{ij}$is a diagonal degree matrix W^l is a layer-specific weight matrix, and $\sigma(\cdot)$ is an activation function (ReLU). H^l is the matrix of activations at the lth layer and $H^0 = X$. The GCN model is built by utilizing multiple convolutional layers. While GCN was not designed for chemical structures, the model is adaptable to them.[228]

One limitation of GNNs is that they can only make predictions for single graph at a time, meaning that complex systems, such as those with separate molecules for solutes and solvents, cannot be handled by standard GNNs. A solution to this limitation is to use a post-processing, mid-model representation of the graph as a fixed-sized vector. AEs are

NNs that compress data into a lower-dimensional, intermediate representation called a latent vector, and then reconstruct the original data as their target output from the latent vector.[239–241] The compression is learned using an encoder, and then a decoder reconstructs the original data from the latent vector.[166,239–242] With an AE, the encoder achieves data compression by successively passing data through NN layers that typically reduce the dimensionality of the data at each layer. The encoder layers are followed by decoder layers that expand the data back to its original form, with both the encoder and decoder learning how to improve their respective parts of the reconstruction. The latent vectors correspond to the output of the smallest, most compressed layer, which is the output of the encoder and thus, situated between the encoder and decoder. While AEs have an intermediate representation in the form of the latent vectors, standard AEs cannot be used with molecular graphs.

A more advanced class of AE, the VAE can utilize data with heterogeneous dimensionality, such as molecular graphs. VAEs extend this concept by modeling the latent space as a probability distribution, enabling more robust and flexible representations.[166,221,242] Specifically, the encoder predicts the mean and variance for each dimension of the latent space, allowing the model to sample from this distribution during reconstruction. Using a probabilistic approach enables more flexible and robust representations of the input data. A VAE is advantageous over a conventional AE because it models the latent space as a distribution, allowing for more flexible and robust representations. The values in latent space are then sampled from this distribution and used

to reconstruct the original inputs. In a VAE, the encoder outputs the mean (μ) and covariance matrix $C(\cdot,\cdot)$ of a normal (Gaussian) distribution: $z \sim \mathcal{N}\left(\mu, C(\cdot,\cdot)\right)$, which is then used to sample the latent representation. One advantage of VAEs over standard AEs is their ability to handle diverse data types, such as molecular graphs. To utilize graph data with VAEs, GNNs can be incorporated into the encoder of the VAE, as demonstrated in the framework shown in Liu et al.[166]

The encoder employs a gated graph neural network (GGNN) to map each molecular graph into a lower-dimensional latent space. Unlike standard GNNs, GGNNs incorporate node labels as additional features, enriching the representation. For each node in the input graph, the GGNN outputs the parameters (mean and standard deviation vectors, μ_v and σ_v) of a d-dimensional diagonal normal distribution, thereby defining the latent space. The latent vectors z_v are then sampled from these learned distributions. During training, the Kullback-Leibler (KL) divergence between the learned and prior distributions is used as a regularization term. The decoder reconstructs the molecular graph by iteratively adding one node or edge at a time, constrained by a predefined maximum graph size.

To reconstruct the graphs at the decoder stage, first, the node states $h_v^{t=0}$ are initialized as a set of unconnected nodes for each graph G in the dataset. Then, sampling each node specification z_v for each node v. Then, each node label τ_v is generated using the learned mapping $\tau_v \sim f(z_v)$. The function $f(\cdot)$ determines the atom type (C, N, O, F, etc.) and its ions, with f being represented by a NN. The probability of reconstructing the labels

τ_v^* is computed by considering all possible permutations of node assignments. During training, the model learns the sequence of edges by performing a breadth-first search (BFS) traversal of each graph. The BFS explores all edges in the current node before proceeding to the next level. Because training over all possible BFS traces is computationally infeasible, the BFS uses approximations of this expectation using Monte Carlo (MC) sampling of a subset of node traces. Then the expected log-likelihood of all traces Π, which serves as a lower bound to the log-likelihood of generating each graph, is calculated. Then each trace is decomposed into a sequence of steps using node v and edges ε added at the current step t. The traces π can be described as shown in equation 1.16,

$$\log p\left(\pi \middle| G^{(0)}\right) = \sum_{(t,v,\varepsilon)\in\pi} \{log\ (p(v|\pi,t)) + \log(p(\varepsilon|G^{(t-1)},v))\} \tag{1.16}$$

where the first term: $\log\ (p(v|\pi,t))$ represents the choice of node v as the focus node at step t of trace π. The second term is conditioned on the current graph. The VAE is trained by minimizing the reconstruction loss, maximizing the log-likelihood of generating the observed graph from the initial unconnected nodes. Finally, the graphs are reassembled by iteratively adding the edges to the unconnected graph to form the molecule that a valid SMILES can represent.

To ensure that the generated graph represents a valid molecule, a valency mask is applied. Each node type is determined by the function f, which has a fixed number of bonds $\mathrm{b}_{\mathrm{v}}^{*}$, given by known properties for each node type. During generation, masks M and m ensure that the number of bonds b_{v} at each node does not exceed the number of bonds for that node type. At the end of the graph generation, $\mathrm{b}_{\mathrm{v}}^{*} - \mathrm{b}_{\mathrm{v}}$ hydrogen atoms are added to each node v as needed to satisfy valency. Finally, the SMILES string of the generated molecule is constructed from the completed graph using the RDKit parser.[198] Unlike a standard VAE, which utilizes solely reconstruction and KL losses as its loss function, it uses reconstruction (sum of node and edge losses) and KL losses with a constraint function that ensures that only valid molecules are generated; this is not considered part of the loss function.[149] The specific loss functions are shown in equations 1.17-1.20.

Each term in the loss function has a specific purpose: the node loss trains the model to predict the nodes (atom types), and the edge loss trains the model to predict the edges (bonds and bond types) between the atoms. The KL loss regularizes the latent representations, ensuring they follow a normal (Gaussian) distribution, maintaining proper sampling and smoothness. The subterms in each part of the loss function are as follows: N, the number of nodes in the graph, C, the number of possible node types (atoms), y_{ic}^{node} the one-hot vector indicating that node i is class c (1 if yes and 0 if not), $\hat{y}_{ic}^{node}$ is the predicted probability for node *i* being class c, which is the output of the SoftMax function, *t* indexes the steps in the edge generation sequence, y_t^{edge} is the true edge label at step t, $\hat{y}_t^{edge}$ is the predicted edge label at step t, $mask_y$ is a binary mask to ignore padding or invalid

steps during generation, H is the dimensionality of the latent space per node, μ_{ik} and σ_{ik}^2 are the mean and variance for the latent dimension k of node i output by the encoder, $mask_i$ is the mask for padding nodes, if present, and finally, λ is a scalar unit that controls the balance between the reconstruction loss and KL loss.[166]

For imaging such as STEM, a CNN model with an AE-like architecture can be used to segment microscopy images into a set of labels, i.e., substrate particles. Mask R-CNN leverages CNNs to extract high-dimensional feature representations from images.[243,244] However, rather than reconstructing the input image as in an AE, mask R-CNN focuses on object detection by predicting bounding boxes and class labels for distinct regions within an image. To achieve this, R-CNN first generates a set of region proposal candidates likely to contain objects using algorithms such as selective search. In this case, each particle has its own matrix, allowing the model to handle overlapping particles individually and adapt to images with varying numbers of objects. Each proposed region is then resized and passed through a CNN to extract features that are subsequently classified and refined by separate fully connected layers. The approach used by R-CNN enables the model to detect and localize multiple objects, including those from the same class (e.g., background or nanoparticles), within a single image. Such capability is particularly valuable for complex visual analysis in STEM, where accurate differentiation between densely packed or touching particles is essential. The original R-CNN structure comprises three main stages: (1) region proposal generation, (2) feature extraction via a CNN encoder, and (3) region classification and bounding-box refinement.

A more traditional representation of molecular data is provided by molecular descriptors, such as those generated by RDKit and PaDEL. Molecular descriptors are represented as fixed-sized vectors with one index per descriptor. Molecular descriptors can be used with a wider variety of ML models, including GTB, linear, and Gaussian models. Three types of GTB are studied: XGBoost, LightGBM, and CatBoost. XGBoost is a variant of GTB and is unique in how it uses GTB, taking functions as parameters. [180,245] XGBoost is an optimized GBM variant that enhances training efficiency through parallelization and regularization. [180,245] It builds an ensemble by sequentially adding trees, where each new tree corrects the residuals of the previous ensemble using second-order gradient information (first- and second-order derivatives of the loss). XGBoost offers key advantages over standard GBMs: (1) built-in L1 and L2 regularization to prevent overfitting, (2) native handling of missing data by learning optimal split directions, and (3) support for parallel feature splitting to accelerate training. The algorithm proceeds by initializing a prediction, computing gradients and Hessians for the current ensemble, finding optimal splits via histogram approximation, applying regularization to prune insignificant nodes, and updating predictions with a learning-rate-controlled step, iterating until convergence or reaching a maximum depth.

LightGBM accelerates GBM training through gradient-based one-sided sampling (GOSS) and exclusive feature bundling (EFB).[246] GOSS prioritizes instances with large gradients (retaining them in full) while randomly undersampling those with small gradients (and scaling their contributions), ensuring that both gradient magnitudes influence the

model equally without altering the overall data distribution. EFB further reduces dimensionality by grouping mutually exclusive sparse features (e.g., one-hot encodings) into single bundled features, making it particularly effective for high-dimensional, sparse datasets such as molecular descriptors.

CatBoost emphasizes robust handling of categorical features (such as those in Morgan fingerprints) and prevention of target leakage.[247] It employs ordered boosting, which computes gradients using random permutations of the training data to avoid bias from future samples seeing past labels. CatBoost constructs symmetric trees for consistent splits across depths, processes categorical variables natively via target statistics (without one-hot encoding), and incorporates feature interactions to boost generalization on mixed numerical-categorical datasets.

In addition to GTB models, the Gaussian Process (GP) model was also studied, providing uncertainty estimations without the need for gradient descent. Gaussian processes (GPs) model functions as draws from a multivariate normal prior, enabling predictions $f(\cdot)$ such that $f(\cdot): x \rightarrow y$, with uncertainty that accounts for error and noise.[148,160,248] A GP model explicitly quantifies uncertainty, making it well-suited for small datasets and cases where probabilistic predictions are essential. A GP model is mathematically represented as shown in equations 1.21 and 1.22,

$$f(\cdot) \sim GP_M\big(h(\cdot), C(\cdot,\cdot)\big) \tag{1.21}$$

$$y_n \mid f(\cdot) \sim Normal_M(f(x_n), v_n), v_n = \epsilon_n^2 \tag{1.22}$$

where M is the dimensionality of the data, $h(\cdot)$ is the mean function of the Gaussian prior, typically set to zero. The covariance kernel, $C(\cdot,\cdot)$, captures relationships between data points.[160] The model is trained using input data points $x_n \in X$ and corresponding target values $y_n \in Y$, where the observed outputs are expressed as $y_n = f(x_n) + \epsilon_n$. The term ϵ_n represents the standard deviation of each observation, modeling inherent noise.

The GP covariance kernel determines the model's predictive behavior. Commonly used covariance kernel functions include quadratic, exponential, rational quadratic, white noise, and linear combinations of multiple kernels. In this work, the quadratic exponential kernel was used, which is mathematically expressed in equation 1.23:

$$C(x, x') = \lambda^2 \, exp\left(-\frac{(x - x')^2}{2l^2}\right) + \alpha^2 \tag{1.23}$$

In equation 1.23, the shape parameter λ controls the variance of the predictions, while the length scale parameter l determines the smoothness of the fitted function. Both

parameters influence the model's tendency to overfit the training data, thereby enabling better generalization. The term α represents a noise parameter, while x and x' are arbitrary data points from the input space X.

A Gaussian prior distribution is defined by its mean function $h(\cdot)$ and covariance kernel $C(\cdot,\cdot)$. Once the training data (x_n, y_n) is observed, the prior distribution is updated to form a posterior distribution obtained through a Bayesian Optimization Method. The posterior remains a Gaussian distribution, allowing predictions to be made for a set of unknown data points $(x*, y*)$ conditioned on the observed data and the covariance kernel. GPs scale slower than simple models ($O(n^3)$ from matrix inversion) but offer uncertainty without gradients.

In addition to making predictions like other ML models, GPs can also be used to estimate the optimum of a black-box function. Bayesian optimization (BO) is a powerful tool that enables efficient exploration of a search space without relying on exhaustive search methods.[249–251] Initially, the BO samples the function at n pseudo-randomly selected points and records the corresponding outputs. Using these evaluations, the BO constructs a GP surrogate model that estimates the expected output and associated uncertainty across the entire search space. In each of the following m iterations, BO selects the point with the highest uncertainty, weighted by the expected loss at that point, and trains the GP surrogate model at that point, updating it with each new observation using the lower confidence bound (LCB) as the search metric.

Although only $n + m$ points are directly evaluated, selecting the m high-uncertainty points allows the BO to approximate the entire search space efficiently. While the selected point is near-optimal rather than the guaranteed global minima obtainable via an exhaustive search, the uncertainty-aware nature of BO allows quantification of the discrepancy between the selected point and the true optimum.

A loss function is a function that ML models use to evaluate their performance during training. It quantifies the difference between the model's predictions and the true values. The goal of training is to minimize this loss using an optimization algorithm, typically a variant of gradient descent, by adjusting the parameters of the model. In essence, the loss function acts as an error metric that guides the model toward better predictions. Common loss functions for classification tasks include CE loss and its variants, and for regression, L1 and L2 losses, also referred to as MAE and root mean squared error (RMSE), respectively.[252–254] Although there are many other loss functions, each with its own variants. Loss functions can also be customized for specific tasks, such as those used by physics-informed neural networks (PINNs), making them more powerful for solving tasks governed by the laws of physics and ensuring only proper outputs and those better governed by real-world physics.[255]

CE loss, also called log-loss, is used to evaluate the performance of classification models whose outputs are probability distributions over $[0, 1]$ for each class. The loss decreases as the predicted probability distribution approaches the true distribution, which assigns a value of 1 to the correct class and 0 to all other classes.

$$CE = -(ylog(p) + (1-y)\log(1-p) \in [0, \infty) \quad (1.24)$$

$$CE = -\sum_{c=1}^{M} y_{o,c} \log(p_{o,c}) \in [0, \infty) \quad (1.25)$$

In equations 1.24 and 1.25 M is the number of classes, y is the true label (binary 0 or 1 for binary classification, or one-hot encoded for multi-class), p is the predicted probability of the observation belonging to a given class, log is the natural logarithm (ln), o indexes observations and c indexes classes. CE loss and its variants are the most commonly used loss functions for classification tasks. Its variants include binary CE and the widely used sparse categorical CE.

L1 and L2 losses, also known as MAE and mean squared error (MSE), respectively, are commonly used loss functions for regression tasks. L1 loss (MAE) measures the average absolute difference between the predicted values and the true values shown in equation 1.26, and L2 loss (MSE) measures the average squared difference between the predicted values and the true values, which is useful to minimize large outlier losses and is shown in equation 1.27.

$$MAE = \frac{\sum_i^n |y_i - \hat{y}_i|}{n} \in [0, \infty) \tag{1.26}$$

$$MSE = \frac{\sum_i^n (y_i - \hat{y}_i)^2}{n} \in [0, \infty) \tag{1.27}$$

In equations 1.26 and 1.27, y_i is the true value, $\hat{y}_i$ is the predicted value, and N is the number of observations. The key difference is that MSE penalizes larger errors more heavily than MAE, making it more sensitive to outliers, while MAE treats all errors linearly.

All of the models discussed, including variants of NNs and GBTs, except for GP models, use a form of SGD.[134,256–261] Standard SGD is shown in equation 1.28 with backpropagation shown in equation 1.29, with the gradient update for the weights in equation 1.30.

$$\theta^{t+1} = \theta^t - \alpha \nabla_\theta J(\theta^t, x^i, y^i) \tag{1.28}$$

$$\delta^{(l)} = (\theta)^T \delta^{(l+1)} \odot (z^{(l)}) \tag{1.29}$$

$$\frac{\partial \mathcal{L}}{\partial \theta^{(l)}} = \delta^{(l)} (a^{(l-1)})^T \tag{1.30}$$

In equation 1.28, θ^t is the model parameters at the previous step with θ^{t+1} being the parameters at the current step, α is the learning rate that determines the rate of change at each iteration, typically starting at a larger value (~0.0001-0.1) to avoid local minima. However, these values vary widely across models and SGD variants. The learning rate then decreases either based on a set number of iterations or on the current improvement of the model performance, $J(\theta^t, x^i, y^i)$ represents the loss function for the parameters θ^t at iteration t with respect to an individual data point (x_i, y_i), ∇_θ is the set of partial derivatives of the loss function with respect to each element of the gradients. The advantage of backpropagation over standard gradient descent is that it updates model gradients layer by layer rather than in a single large step. In equations 1.29 and 1.30, $\delta^{(l)}$ is the error term at layer l, $\theta^{(l+1)}$ is the weight matrix for the next layer, $z^{(l)}$ is the weighted input of layer l, $\mathcal{L}$ is the loss function, and $a^{(l-1)}$ is the activation function of the previous layer.

The most common variant of SGD used in NNs is the Adam optimizer designed by Diederik et al and described in Algorithm 1 in Diederik et al.[256,261] Adam is preferable over SGD as it is better optimized for NN models because it adapts the learning rate per parameter, leading to faster and more stable training that is less sensitive to fine-tuning the learning rate. Non-NN models such as GTBs use standard SGD, unlike the SGD variants used in NNs, where the optimization space comprises the model gradients; in GTBs, the optimization space comprises possible splits, aiming to find optimal tree splits rather than NN gradients. In a real-world scenario, such as NNs or GBTs, the search space will be

larger. The same phenomenon becomes more pronounced: the loss landscape contains many basins and saddle regions, so different random initializations can place SGD in very different regions of this landscape, often leading to convergence toward distinct local minima or flat valleys, which strongly influences both optimization speed and generalization behavior. Rather than typical pseudo-random initialization, more methodical approaches such as Xavier initialization are helpful; the performance gain is minimally affected by data processing and model structure, far more by them. [262]

Several metrics are commonly used to evaluate model performance. For data with discrete or categorical values, metrics such as accuracy, precision, recall, confusion matrices, the receiver operating characteristic (ROC) curve, and the area under the curve (AUC) are frequently used. For continuous data, commonly used metrics include MAE, root mean squared error (RMSE), and the R-squared (R^2) score. Using multiple metrics together provides a more holistic understanding of an ML model's performance. Other approaches include percentage-based error metrics, mean squared error (MSE), and deviance-based errors. Beyond these, numerous variants and application-specific metrics exist, such as graph similarity measures for evaluating models on graph-based data.

These methods integrate diverse molecular representations, from SMILES and RDKit descriptors to ASE graphs, CMs, and ECFPs, using advanced ML architectures spanning GTBs, GPs, GNNs, VAEs, and image detection NNs, supported by CV and uncertainty quantification. Collectively, these approaches create a reproducible framework that utilizes ML for scientific discovery across chemical and materials systems.

The work presented in this dissertation leverages ML to address critical bottlenecks in RFB electrolyte design and electrode analysis, targeting five key properties: EACN, CMC, MP, non-aqueous solubility, and nanoparticle distributions in PCEs. By developing GNNs, GTBs, GPs, VAEs, and Mask R-CNN models, this work enables high-throughput screening of μE components for the type III and IV phases, and automates STEM image segmentation for precise nanoparticle characterization. These approaches help overcome experimental limitations in throughput and scale, accelerating BEES2 EFRC goals for high-energy-density electrolytes and improved energy storage performance.

CHAPTER 2:

COMPARISON OF MACHINE LEARNING APPROACHES FOR PREDICTION OF THE EQUIVALENT ALKANE CARBON NUMBER FOR MICROEMULSIONS BASED ON MOLECULAR PROPERTIES

Abstract

The chemical properties of oils are vital to the design of μE systems. The HLD equation used to model μE phase behavior expresses the oil's physicochemical properties in terms of the EACN. The experimental determination of the EACN requires knowledge of the temperature dependence of the μE system and the effect of different surfactant concentrations. Thus, the experimental determination is time-intensive and tedious, requiring days to months for proper separations. Furthermore, the experiments require high chemical purity because μEs are sensitive to impurities. The work in this chapter focuses on quick, reliable predictions of the EACN using several ML models. Four ML models were compared: three GNNs and XGBoost. The GNNs used molecular structures as the initial input and assessed whether geometry optimization is necessary for reliable results. The XGBoost model used molecular descriptors and RFECV to find the optimal descriptor set.

Previously published as: Furth, N. R., Imel, A. E., Zawodzinski, T. A. (2024). Comparison of Machine Learning Approaches for Prediction of the Equivalent Alkane Carbon Number for Microemulsions Based on Molecular Properties. The Journal of Physical Chemistry A, 128(32), 6763–6773. https://doi.org/10.1021/acs.jpca.4c00936.

Introduction

Electrochemistry in μEs has recently seen renewed interest due to the use of μEs as electrolytes in energy storage devices, such as RFBs.[5,6,27,33,263,264] μEs are thermodynamically stable, homogeneous solutions of polar and non-polar molecules with a surfactant stabilizing the immiscible liquids. μEs often contain additives, such as salt or cosurfactants, that optimize properties for specific applications. The stabilization of mixtures of nanoscale polar and nonpolar phases in μEs is important for their use as alternative electrolytes, as they can accommodate organic redox species (not soluble in aqueous solvents) while maintaining high conductivity comparable to that of aqueous electrolytes. Decoupling ionic conductivity from the solubilization and transport of redox-active species enables separate optimization of power and power density. There is a significant base of experience with μEs to draw from based on their applications in nanoparticle and polymer synthesis,[265–267] and oil recovery.[268–270] By 1948, Winsor introduced the concept of relative interaction energies of the surfactant at the oil-water interface and in the respective oil or water phases.[271] The resulting Winsor system describes the structural phases of μEs with four distinct types. Type I μEs have discrete oil droplets in a continuous water medium, and type II consists of discrete water droplets in a continuous oil phase. Type III μEs are bicontinuous, meaning both the non-polar and polar phases are continuous, with excess water and oil. In type IV, the oil and water are homogenous in a single phase, typically with a bicontinuous morphology. A μE with an optimum formulation in the Winsor system is a bicontinuous type III or type IV μEs, which

form when the chemical potential of the surfactant in the excess aqueous and oil phases is equal, corresponding to the minimum interfacial tension.[272–276]

By the 1970s, Salager presented the empirical HLD equation to improve the existing hydrophilic-lipophilic balance (HLB) equation.[273] The HLD equation is a dimensionless linear relationship that includes all the formulation variables, such as EACN, Cc, temperature, and salinity. The HLD equation predicts optimum formulation at HLD ≈ 0, forming a type-III or type-IV μE in the Winsor system. Furthermore, the HLD equation, which considers the water and oil phase compositions and the temperature, surfactant, and cosurfactant, can predict the Winsor classification. The HLD equations are shown in equations 1.1 and 1.2 for both ionic and non-ionic systems. While these are empirical equations, each term has a basis in a physical phenomenon.[277] These terms are discussed in reference to Wang et al.[277] The salt concentration (S), in wt% of the system, accounts for suppression of the ionic surfactant electrical double layer by ion pairing, while b is an empirical constant reflecting the maximum solubility of surfactant in the electrolyte.[278] The functions $f(A)$ and $\phi(A)$ vary with the type and concentration of the cosurfactant. If there is no cosurfactant, $f(A)$ and $\phi(A)$ are 0.[275] The change in temperature from 298 K is ΔT, with values of $\alpha_t = 0.01\,K^{-1}$ and $c_t = 0.06\,K^{-1}$ commonly reported in the literature.[272,276,279] The EACN is a numerical coefficient that reflects the polarity of the oil.[276,280] The constant K typically ranges from 0.15 to 0.17, which scales the EACN. Finally, the variable Cc represents the surfactant's hydrophilic nature and the μE equilibrium time.

The EACN has several effects beyond the HLD equation, including hydrophobicity, Winsor IV stability, oil-loading capacity, and interfacial tension.[28,54,66,71,281,282] The EACN is influenced by the interactions between the surfactant tail and the oil molecules. It describes the oil phase hydrophobicity and its ability to penetrate the interfacial film and modify the spontaneous curvature.[275,276,283,284] The EACN represents complex nonpolar molecules as straight alkane carbon chains. However, beyond those, there is no intuitive basis for estimating EACN at a glance. For example, hexane and isohexane are structural isomers with different branching but the same molecular formula, and have different EACN values. Hexane has an EACN of six, while p-menthane has an EACN of 5.8.[285] Additionally, unlike straight carbon chains, some molecules can have negative or fractional values of EACN, such as cyclohexene with an EACN of -1.2.[285] Thus, for more complex oils, with cyclic or branched hydrocarbon moieties, the EACN needs to be experimentally determined.

Evaluation of oil hydrophobicity using the EACN is essential in many industries, as it guides surfactant selection. For instance, eliminating greasy stains from fabric,[286] removing cosmetics,[287] perfumery,[288] or extended oil recovery in the petroleum industry[289] rely on understanding the oil to form optimum μEs. Reliable measurements of EACN are complex and require extensive work and a pure substance≥ 98%.[70,73,74] Additionally, the choice of surfactant when determining the EACN can affect the result. For this reason, predicting EACN using only oil properties with ML is difficult. For example, a short-chain surfactant can dissolve in either the water or oil phase, and free, unaggregated surfactants

are likely present in both phases. Therefore, a longer-tail group, about 12 carbons, that dissolves less readily would be preferable, but many $C_{12}E_j$ surfactants require several days to weeks to equilibrate in μE environments.[276] Furthermore, for more polar oils, with EACNs of less than six, the calibration curves established with n-alkanes have more significant uncertainty. Therefore, experimentally measuring the EACN requires determining the optimal temperature T* of the μE using a fish-tail temperature plot, which is tedious and time-consuming because the process must be repeated for each new oil.[72,73]

Previous work focused on applying μE systems as part of BEES with a standard redox-active molecule (ferrocene) dissolved in toluene as the oil phase[5,6,27,33,263,264] These studies demonstrated that μEs could serve as electrolytes in energy storage devices but highlighted the need for more concentrated redox-active non-polar phases to maximize power density. With each new redox-active oil tested, a new pseudo-ternary phase diagram is needed. However, the HLD equation has shown promise as a screening method for gaining insight into the type of μE formed with new oils, thereby circumventing the need for extensive experiments. In this way, obtaining reliable EACN values is of considerable value in developing new electrolyte formulations.

The use of ML in chemical systems ranges from molecular structure inputs to hybrid computational-ML approaches that combine ML and computational methods, to more traditional descriptor-based approaches.[92,147,216,290–295] The work in this chapter uses a small database of 186 reported EACN values to train and compare state-of-the-art GNNs, with molecular structures as inputs, and to train and compare XGBoost with molecular

descriptors as inputs.[180,245] Because the GNNs use molecular structures as the inputs, this chapter also studies how geometry optimization affects the predicted values from the NNs.

The ML models studied can accurately predict EACN from molecular graphs. Several computational methods now predict EACN directly from molecular structure. Lukowicz et al. introduced a COSMO-RS approach using surface charge density σ-moments (σ_0, σ_2) as descriptors, accurately capturing polarity-driven deviations from nominal carbon number. Building on this, Delforce et al. developed two rapid models: a GM trained on SMILES strings and an NN using COSMO-RS σ-moments, both showing comparable accuracy, with the GM better capturing homologous trends and the NN leveraging quantum features. Acree et al. correlated EACN with Abraham solute parameters using five transferable descriptors, thereby enabling high-throughput screening. At the macroscopic scale, Yoga et al. applied the HLD-NAC model with pressure corrections to obtain physically consistent EACN values for microemulsion and EOR formulations.

Methods

A database of 186 organic molecules with known EACNs was compiled from the literature.[69,70,72,73,197,296,297] The molecular structures were used as inputs, first converted into their SMILES by querying the PubChem database[197] or drawing the molecular structure using cheminfo.org[298] and converting it to the SMILES format.

Figure 2.1 illustrates the procedure for data processing and model training. Following the upper path in **Figure 2.1**, using the Open Babel program to produce xyz files.[299] The xyz files describe molecular geometries, either non-optimized or optimized with MMFF94 or GAFF. The non-optimized geometries are only two-dimensional, making them, in essence, xy representations, whereas the optimized geometries are in three dimensions, and the xyz representations are then converted to a JSON format.[224,231,232,300] Using the data in JSON format, the structures are converted to sparse bond-connection lists (inferred from xyz coordinates using RDKit with >99% accuracy), which replace full adjacency matrices, creating memory-efficient graph representations. During this stage of data processing for MatDeepLearn, three of the 186 graphs failed to process, leaving 183 data points for the GNNs. In MatDeepLearn specifically, atom feature vectors contain only the one-hot-encoded atomic number and one-hot-encoded coordination number (neighbor count within the cutoff), while bond feature vectors encode interatomic distances via a Gaussian radial basis expansion. While atom features remain identical between optimized and non-optimized geometries, bond features differ due to geometry-dependent atomic distances.[301] For relation to EACN, the one-hot encoded atomic number distinguishes C/H atoms from heteroatoms as more carbons increase EACN through enhanced chain length, while heteroatoms disrupt hydrophobicity. Edge connectivity encodes critical molecular topology, including linear vs branched chain architecture, terminal methyl vs internal methylene positioning, and heteroatom proximity to hydrophobic carbons. Bond distances (via Gaussian radial basis expansion) quantify conformational extension; more extended

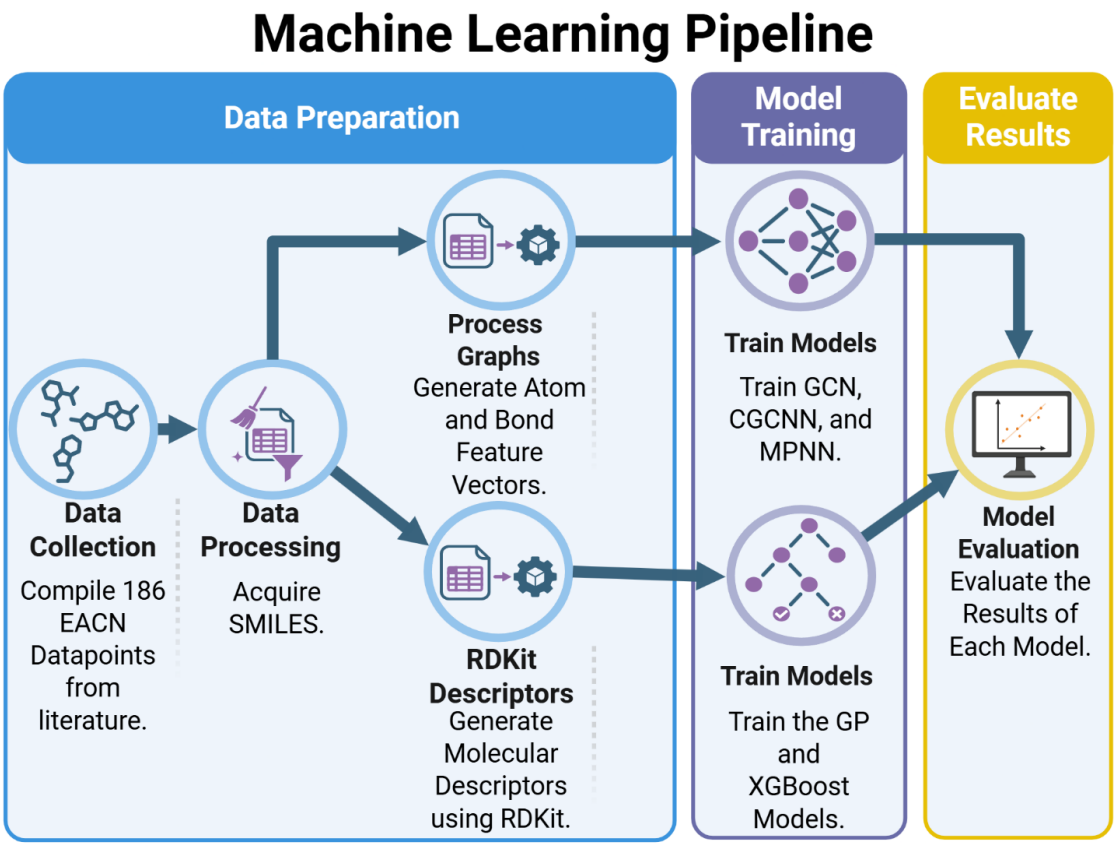

Figure 2.1. ML pipeline for predicting EACN. Created in BioRender. Furth, N. (2026) https://BioRender.com/t0xzjcl

alkane-like C-C distances correlate with higher EACN, while compact conformations reduce effective hydrocarbon character.[52,71,302]

Using MatDeepLearn, an environment used to benchmark various state-of-the-art GNNs, as the framework. Three GNNs were used: MPNN,[238] CGCNN,[168,169,230] and GCN.[222,226,228] A GNN is a specialized class of NN that uses graph data, a non-linear data structure consisting of vertices and edges, of a heterogeneous structure and size, unlike standard NNs, which typically use tabular or matrix data of fixed dimensionality.[223,225,227] A GNN uses graph convolutional and pooling operations to transform input graphs into a fixed-size representation while preserving relevant information. GNNs are common in chemical applications that use molecular structures as inputs.[167,224,225,229] The lower path in **Figure 2.1** illustrates the XGBoost model procedure. Using SMILES as input, the XGBoost model is trained on a set of molecular descriptors generated by RDKit. RDKit uses the SMILES format for inputs and outputs a vector of molecular descriptors, such as topological polar surface area (TPSA), the number of rotatable hydrogen bonds, and the number of rings. There are 244 molecular descriptors; a complete list is in the RDKit documentation, and a detailed description is in **Table A.1**.[215]

In this chapter, the molecular descriptors used for XGBoost were selected using RFECV. RFECV ranks each feature based on its correlation to the target value and the other features. Using the rankings, RFECV then trains a user-defined model with the dataset, starting with only the highest-ranked feature, then adding the next highest-ranked feature after each iteration, until the entire dataset has been added. After training, RFECV selects

the optimal number of features, which yields the smallest average error on each CV subset. **Figure 2.2** shows RFECV plots showing the mean test score as a function of the selected features, and the vertical dotted line indicates the optimal number of features (50).

The descriptors generated utilizing RDKit are too numerous to describe individually. However, these descriptors can be simplified into nine categories.[215] There are 2D descriptors such as atom count, bond count, and type, 3D such as molecular volume, surface area, and torsion angles, topological including wiener and Zagreb indexes, electronic and charge descriptors such as Gasteiger charges, energy eigenvalues, hydrophobicity and solubility descriptors including logP (octanol-water partition coefficient, logS (solubility), geometric and structural descriptors such as molecular weight and diameter, polar surface and solvent accessible area, flexibility descriptors such as rotatable bond and ring counts, and pharmacophoric descriptors such as hydrogen bond donor/acceptor counts.

The structure and size category includes heavy-atom count, heavy-atom molecular weight, chi1n, Estate_VSA6, and amw. The polarity category comprises of fr_oxime, Estate_VSA6, fr_phenol_noOrthoHbond, fr_phenol, lipinskiHBD, and NumAromaticRings. The Estate_VSA6 appears in both categories because it captures how the molecule's shape and electronic environment contribute to its chemical behavior and interactions. A brief description of these features is provided in **Table A.1**. The relation to the structure and size, and polarity of the molecule aligns with the findings of Lukowicz et al., which emphasize the importance of van der Waals surface area, encompassing size and

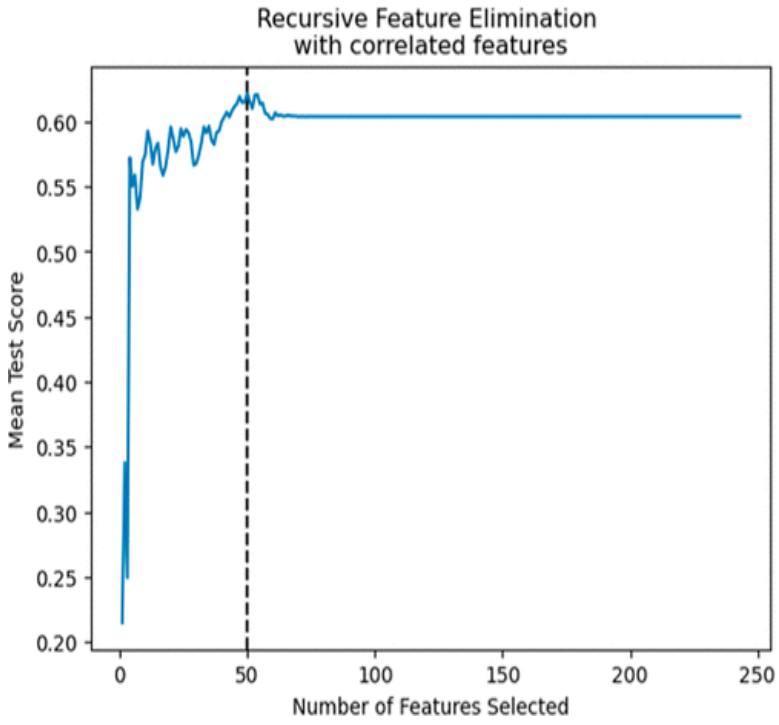

Figure 2.2. RFECV plot for XGBoost, with the vertical dotted line highlights the optimal number of features.

overall polarity, in relation to the EACN.[70] As shown in **Figure 2.2**, only a fraction of these descriptors have a significant correlation with the EACN. Since the EACN, in essence, represents an alternative structural representation of the molecule, descriptors that represent the geometry and structure of the molecule are key.

Both the GNNs and XGBoost have advantages and disadvantages for predicting EACN. Representing molecules as graphs allows the input to use all molecular information without requiring specific properties to be explicitly defined. However, this method is more computationally expensive. Utilizing molecular descriptors enables significantly faster runtime due to their more compact format and the absence of complex operations such as graph convolutions, and they are compatible with a wider variety of traditional ML models than graph data. Additionally, molecular descriptors are more interpretable than graph-based representations. However, descriptors have the disadvantage of requiring careful selection, whether using a broad approach such as RDKit with RFECV or meticulously selecting descriptors that relate to EACN or are easy to obtain experimentally in a high-throughput manner than the EACN, yet still provide a molecular correlation to the EACN.[215]

The work in this chapter also investigates whether molecular geometry optimization is advantageous for GNNs in providing reliable predictions, given the computational resources and time required. Using GAFF, MMFF94, and no optimization as three scenarios for the GNNs. While the GAFF and MMFF94 methods effectively compute optimized molecular geometries, these calculations can be computationally

expensive. The geometry optimization yields more accurate bond lengths, an important factor in predicting EACN.

The outputs of the four models are compared using MAE, which measures the error between predictions and actual values, and RMSE, which measures the precision of the predictions. If all errors have the same magnitude, then RMSE equals MAE, and if they are not the same magnitude, RMSE is larger than MAE. Unlike MAE and RMSE, which demonstrate the model's prediction quality, the R2-score indicates the quality of the fit the model learns, given the variance of the data and the model's errors. To evaluate the effects of different functional groups on the models, an analysis of variance (ANOVA) was used. ANOVA is a method that compares the sums of squares between groups (SSB) and within groups (SSW). SSB measures the variance between different groups, and SSW measures the variance within the groups. Comparing the ratio between SSB and SSW, dividing by $(k-1)$ and $(n-k)$ respectively to calculate the F-statistic as follows in equation 2.1,

$$F-Statistic = \frac{\mathrm{SSB}/(\mathrm{k}-1)}{\mathrm{SSW}/(\mathrm{n}-\mathrm{k})} = \frac{\sum_{\mathrm{i}=1}^{\mathrm{k}} \mathrm{n_i}(\bar{\mathrm{y}}_\mathrm{i}-\bar{\mathrm{y}})^2}{\sum_{\mathrm{i}=1}^{\mathrm{k}}\sum_{\mathrm{j}=1}^{\mathrm{n_i}}(\mathrm{y_{ij}}-\bar{\mathrm{y}}_\mathrm{i})^2}\frac{(\mathrm{n}-\mathrm{k})}{(\mathrm{k}-1)} \in [0,\infty) \quad (2.1)$$

where k is the number of functional groups, n is the total number of data points, n_i is the number of data points in group i, y_{ij} is the error of datapoint j in group $i, \overline{y}$ is the

mean of errors, and $\bar{y}_i$ is the mean error of group i, and j is the index for a data point within group i. The higher this ratio is, the higher the variance is between the errors of the different functional groups, and indicates a higher level of significance.

Results and Discussion

Computational runtime is an important performance metric to determine which algorithm can run within a reasonable time frame.[303] The complexity of GNNs and XGBoost varies with their architectures and parameters. Here, the focus is mainly on comparing the number of parameters for each GNN with its runtime, both of which vary significantly. The CGCNN and GCN models have relatively low parameters, 27,251 and 56,351, respectively. In contrast, MPNN has 1,510,811 parameters, ~25 times as many as GCN and ~55 times as many as CGCNN. A significant number of parameters could drastically affect the models' runtimes, as shown in **Table 2.1**.

As expected, given its substantial number of parameters, MPNN has a significantly longer runtime (~3x-7x) than the other GNNs. GCN has twice as many parameters as CGCNN and, on average, runs ~2.3 times as fast as CGCNN. The faster runtime of GCN is likely due to optimizing the asymptotic runtime (how long the model takes to train relative to the amount of input data), making it more efficient.[73] Both CGCNN and GCN have no significant difference in runtime when comparing the optimized and non-

Table 2.1. Total run time in seconds of MPNN, CGCNN, and GCN with unoptimized vs. optimized geometries using each force field, MMFF94, and GAFF, and XGBoost on data generated using molecular descriptors predicting EACN.

Optimization Method	Model	Run Time (s)
Unoptimized	MPNN	197.2
	CGCNN	68.9
	GCN	28.3
Optimized (MMFF94)	MPNN	212.3
	CGCNN	68.8
	GCN	30.0
Optimized (GAFF)	MPNN	206.6
	CGCNN	67.4
	GCN	29.1
Molecular Descriptors	XGBoost	0.01

optimized geometries, while the MPNN training is slightly faster (<8%) with non-optimized geometries. XGBoost is the fastest model, only needing 10ms to train, which is over 2,900 times faster than the fastest GNN, GCN. While this runtime comparison does not consider the accuracy or precision of the predictions, the variation in runtimes shows that the choice of an ML model can significantly affect training time, underscoring the trade-off between accuracy and speed that ML offers.

Furthermore, the database used in this study is relatively small, comprising only 186 inputs representing molecules that are neither large nor complex. Larger databases and more complex molecules, such as polymers or surfactants, will increase runtimes across all models and exacerbate differences between them.

Three metrics used to evaluate the ML model with the training and testing sets are MAE, RMSE, and R^2. The values are recorded in **Table 2.2** and illustrated in **Figure 2.3**. The data using non-optimized geometries yields the worst non-optimized-versus-optimized predictions across all metrics. A cursory glance at the results is interesting: it shows that providing GNNs with 3D graph data rather than 2D enables them to find correlations that reliably predict EACN values. This is due to the relationship between the EACN and the bond length, which the 2D geometry does not accurately produce. The MPNN model consistently achieves the best results on the training dataset and the worst on the test dataset, indicating overfitting. Overfitting likely occurs because the model is too complex, with too many parameters, to generalize well on such a small dataset. These reasons are further supported by the implementation of early stopping and an L2 regularization term,

Table 2.2. MAE, RMSE, and R^2 for the training and testing datasets for each model evaluated (MPNN, CGCNN, GCN, and XGBoost).

Optimization Method	Model	Training			Testing		
		MAE	RMSE	R^2	MAE	RMSE	R^2
Nonoptimized	MPNN	1.152	1.873	0.935	2.958	4.034	0.537
	CGCNN	1.646	2.288	0.885	2.617	3.379	0.797
	GCN	2.138	3.020	0.828	2.215	3.042	0.735
GAFF	MPNN	0.516	1.044	0.978	1.610	2.559	0.876
	CGCNN	1.187	1.965	0.907	1.504	2.075	0.929
	GCN	1.275	1.823	0.934	1.638	2.124	0.890
MMFF94	MPNN	0.359	0.512	0.994	1.568	2.835	0.837
	CGCNN	1.315	2.060	0.926	1.156	1.551	0.929
	GCN	1.571	2.186	0.887	1.307	1.782	0.927
Molecular Descriptors	XGBoost	0.550	0.970	0.980	1.680	2.280	0.910

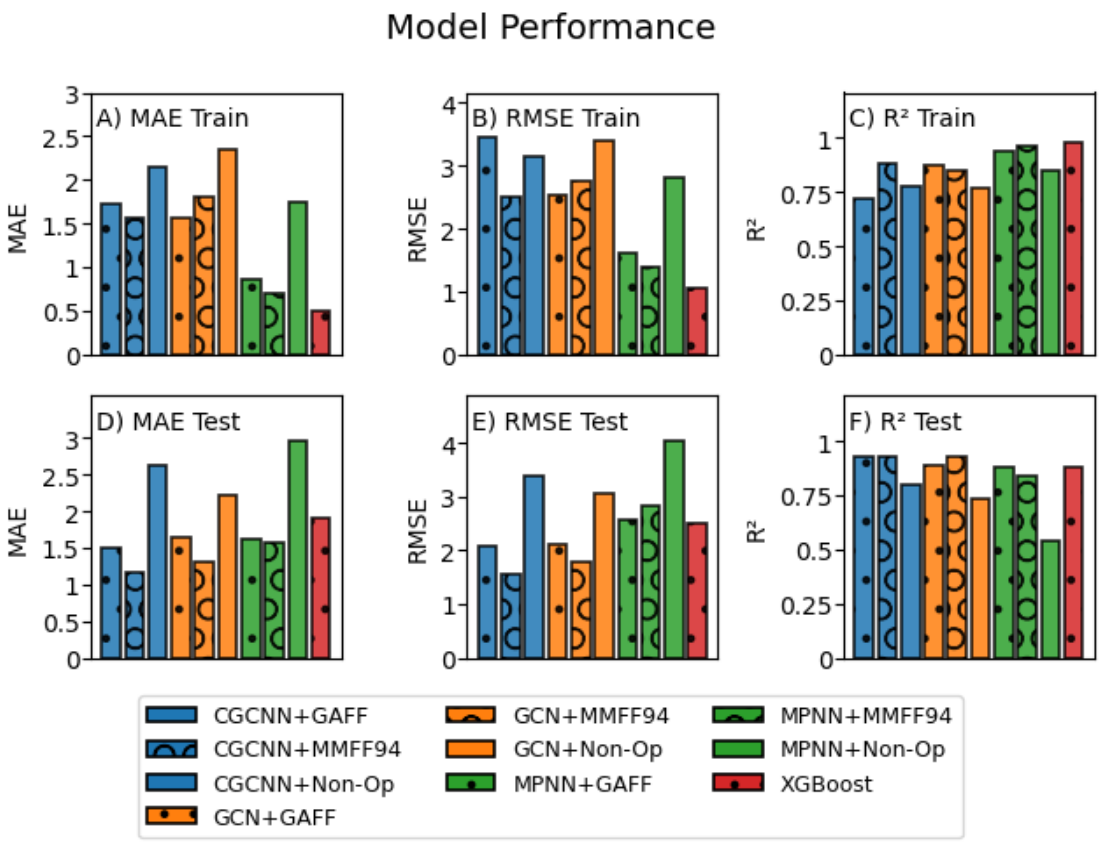

Figure 2.3. MAE, RMSE, and R^2 for the EACN model predictions. (A) Training MAE, (B) Training RMSE, (C), Training R^2, (D) Testing MAE, (E) Testing MRSE, and (F) Testing R^2. Nonoptimized geometries are shown with no hatches, GAFF optimized geometries are shown with dot hatches, and MMFF94 optimized geometries are shown with circle hatches. CGCNN is shown in blue, GCN is shown in orange, MPNN is shown in green, and XGBoost is shown in red.

suggesting the issue is unrelated to individual model gradients. Additionally, early stopping did not activate, indicating that the validation error did not exceed a preset threshold for a sustained period at any point during training. CGCNN and GCN provide substantially better results on the dataset with non-optimized geometries than the MPNN model, consistent with the first reason likely being MPNN overfitting. That is, CGCNN and GCN have a significantly lower number of parameters. XGBoost, the simplest model, performed best on the non-optimized geometries, yielding predictions with significantly lower MAEs and RMSEs than the GNNs.

The MPNN model for the GAFF- and MMFF94-optimized geometries has the same overfitting problem as it did for the non-optimized geometries. CGCNN and GCN achieve good results on optimized geometries, with high R^2 values and low MAE and RMSE. Interestingly, the MMFF94 performs better on the testing dataset, while the GAFF performs better on the training dataset. XGBoost consistently achieves better training results than the MPNN model, but its testing results are considerably worse, indicating overfitting, though not to the degree of MPNN. Overfitting indicates that the model does not generalize well to unseen data. In contrast, the other models do not suffer from overfitting to this degree as they have fewer parameters and can better fit smaller datasets.

The results in **Figures 2.2-2.3** and **Table 2.2** clearly show that molecular geometry optimization significantly affects the accuracy and precision of the ML models. Comparing each of the models and optimization methods at a higher level for the EACN testing set.

For models trained with non-optimized data, the expected error of the predictions is ±2.958 EACN units; for models trained using data optimized with GAFF and MMFF94, the expected error of the predictions is within ±1.638 EACN units, and for the molecular descriptors used to train XGBoost, the expected error is ±1.680 EACN units. The best model tested, CGCNN trained using data optimized with MMFF94, has a much lower error bound, with an expected error of ±1.156 EACN units. While the model predictions are not without error, the errors are sufficiently low to make the models valuable screening tools.

During model training, the training and validation errors were evaluated at each epoch, as shown in **Figure 2.4**. Overall, models trained on non-optimized geometries took longer to converge to a minimum, while the validation set converged early in training. However, both models had larger errors after convergence than models trained with optimized geometries. Models trained on data with optimized geometries exhibited instability in the validation error curves during early epochs (<100), which is common because the validation set is small and cannot fully represent the entire dataset. As a result, until the overall loss converges, the validation loss will be somewhat inconsistent. Both MPNN and CGCNN had smooth training curves, showing little fluctuation. Although converging to a similar point as CGCNN, GCN had larger fluctuations, possibly due to the model being small. Changes in individual weights may have a greater effect on overall performance, leading to larger fluctuations. Finally, MPNN exhibited the greatest instability in the validation error curve, although it converged to the lowest point on both the training and validation sets.

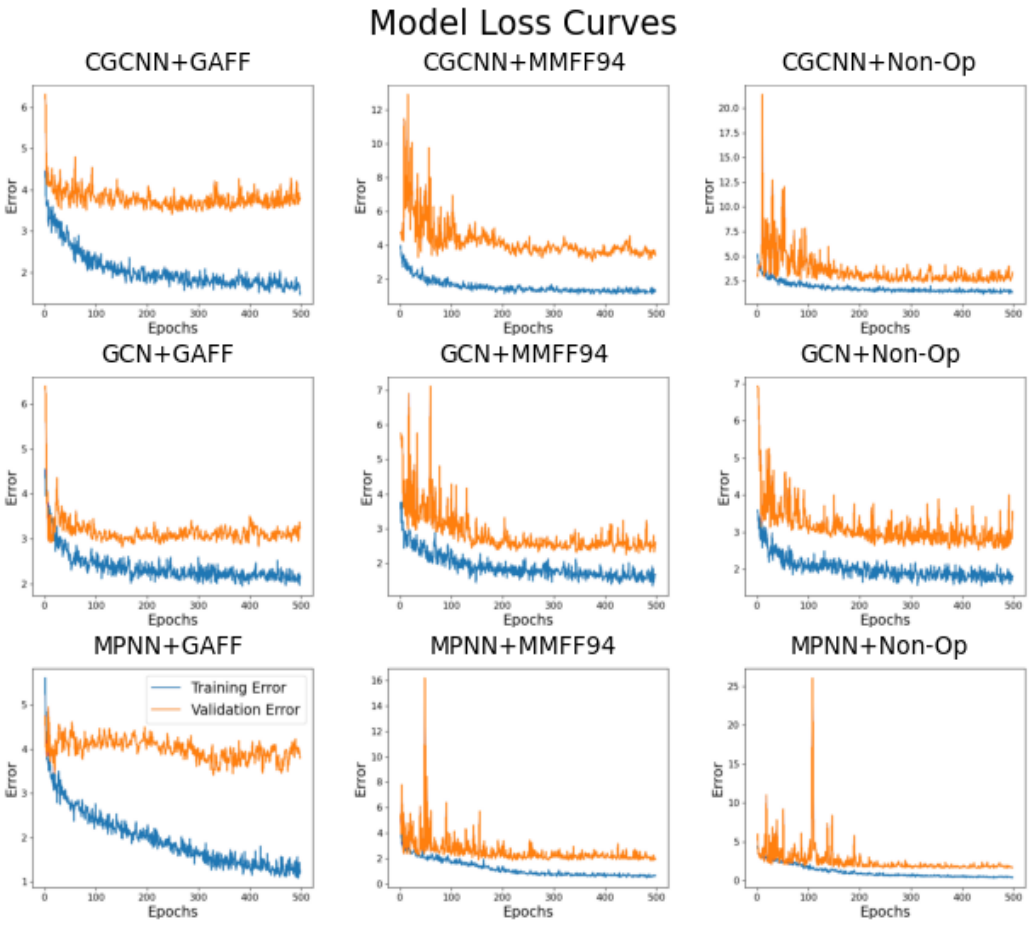

Figure 2.4. GNN model loss curves with training error in blue and validation error in orange.

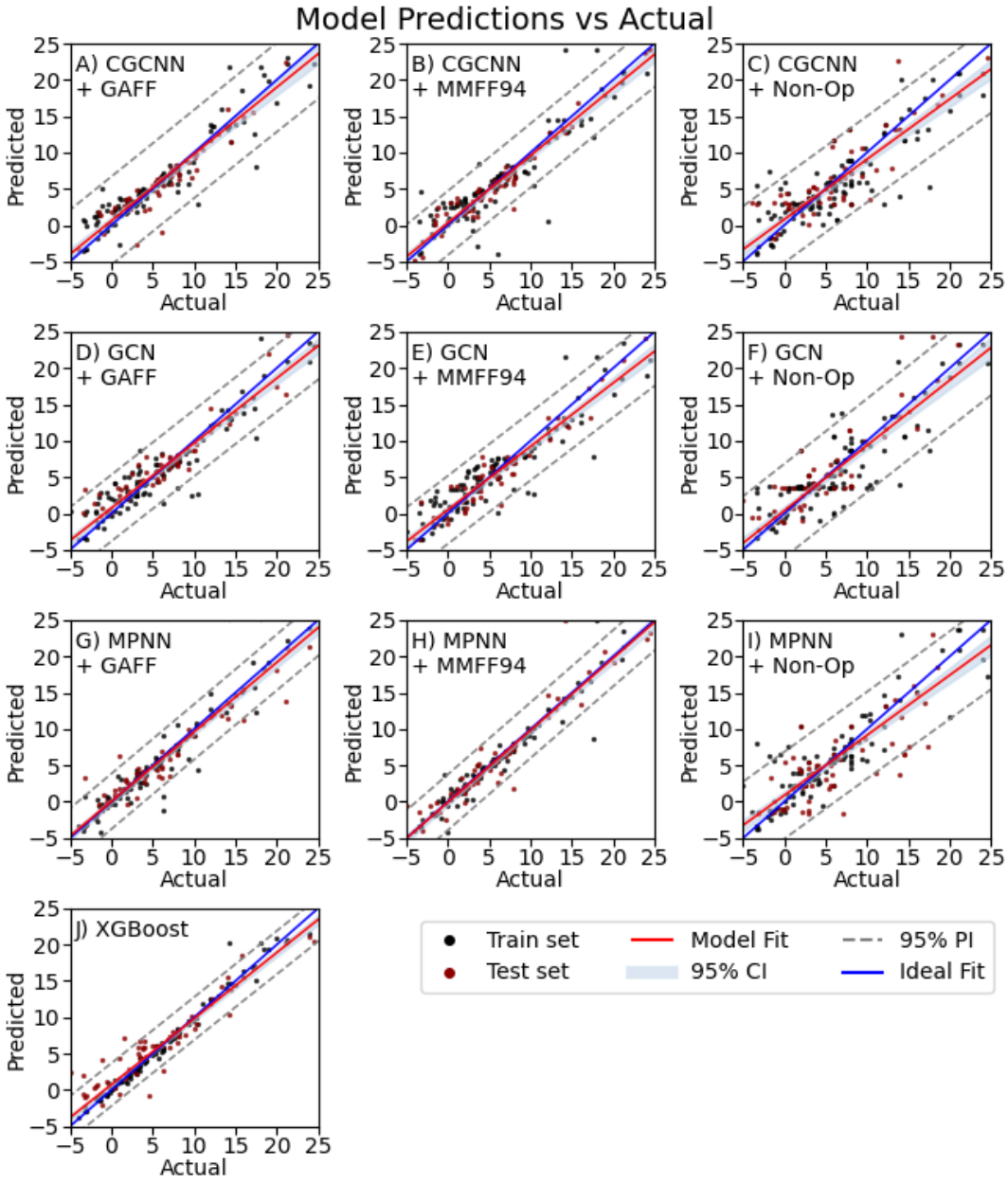

Figure 2.5. Regression plots for each model tested with training prediction shown in black, testing predictions shown in solid red. Ideal fit is shown as a solid blue line and model fit shown in a red line. 95% confidence intervals shown in shaded blue, and the 95% prediction interval shown as gray dashed lines.

In **Figure 2.5**, the values used as target properties are on the x-axis, and the predicted values are on the y-axis. The predicted values are shown in black for the training set and red for the testing set. The fit line is solid red, the ideal fit line is solid blue, the 95% prediction interval is the dashed line, and the 95% confidence interval of the predictions is shaded blue. There is a clear difference between the non-optimized and optimized geometries, with models trained with data using non-optimized geometry having wider prediction and confidence limits than those trained with optimized geometries. Additionally, models trained on non-optimized geometry perform disproportionately worse at very high or very low outlier values. There is little difference between the confidence intervals, confidence limits, and fit lines of models trained using geometries optimized with GAFF and MMFF94.

The MPNN model is more precise overall than CGCNN and GCN; however, this observation is misleading. The model is overfitting and will perform well on most data since the training set is significantly larger than the testing set. The CGCNN and GCN models have similar confidence and prediction limits. However, GCN is more precise overall, as it performs better on outlier values and yields fewer predictions outside the prediction range. The regression plot for XGBoost has the narrowest confidence and prediction limits; however, it still makes worse predictions on outlier values.

Analyzing prediction error distributions for the model shown in **Figure 2.6**, all ten models show a right-skewed distribution, meaning the errors are skewed toward zero. The MPNN model shows a drastic bias towards zero in the training set when trained with data

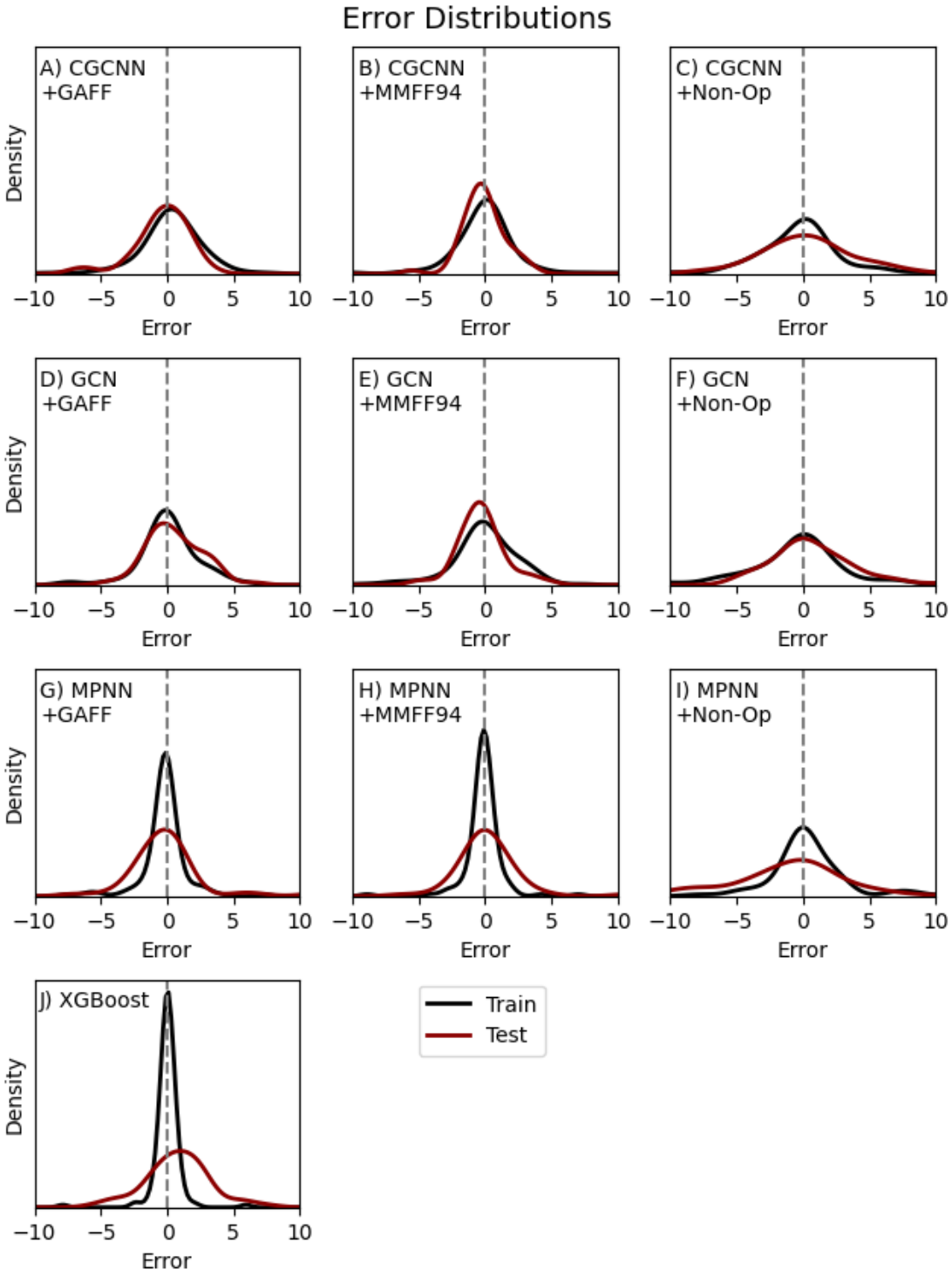

Figure 2.6. Distribution of Errors for the three GNNs with EACN data with different geometry optimization, (A-I) and (J) molecular descriptors with XGBoost with the training set in black and the testing set in red.

optimized with GAFF and MMFF94, and a more moderate skew in the testing sets. When trained with non-optimized data, the training set shows a less significant skew, whereas the testing set appears to be uniformly distributed. The CGCNN model exhibited a uniform distribution across both the training and testing sets when trained on non-optimized data. When trained on data optimized with GAFF and MMFF94, the distributions of the training and testing sets are both skewed toward zero, although the training set has more outliers in the MPNN model. The GCN model showed similar performance to the CGCNN model, when trained with non-optimized data. When trained on data optimized with GAFF and MMFF94, the training errors were less skewed toward zero, whereas the testing errors were more skewed toward zero.

The oils in the dataset were divided into 17 molecular categories listed here with the total amounts in the paratheses, alkane (22), alkene (7), alkyne (7), aromatic (7), cyclic (1), cyclic alkane (23), cyclic alkene (10), cyclic ester (1), ester (20), ether (11), halogenated alkane (21), ketone (5), nitrile (4), substituted aromatic (12), terpene (18), terpenoid (6), and triglyceride (10).

Figure 2.7 shows the MAE (solid) and RMSE (hashed) for the testing dataset points for each model, presented as a set of bar plots. The best predictions came from the esters, ethers, halogenated alkanes, and terpenoids, with the lowest MAE and RMSE consistently. The performance with alkanes, alkenes, alkynes, ketones, terpenoids, and triglycerides was

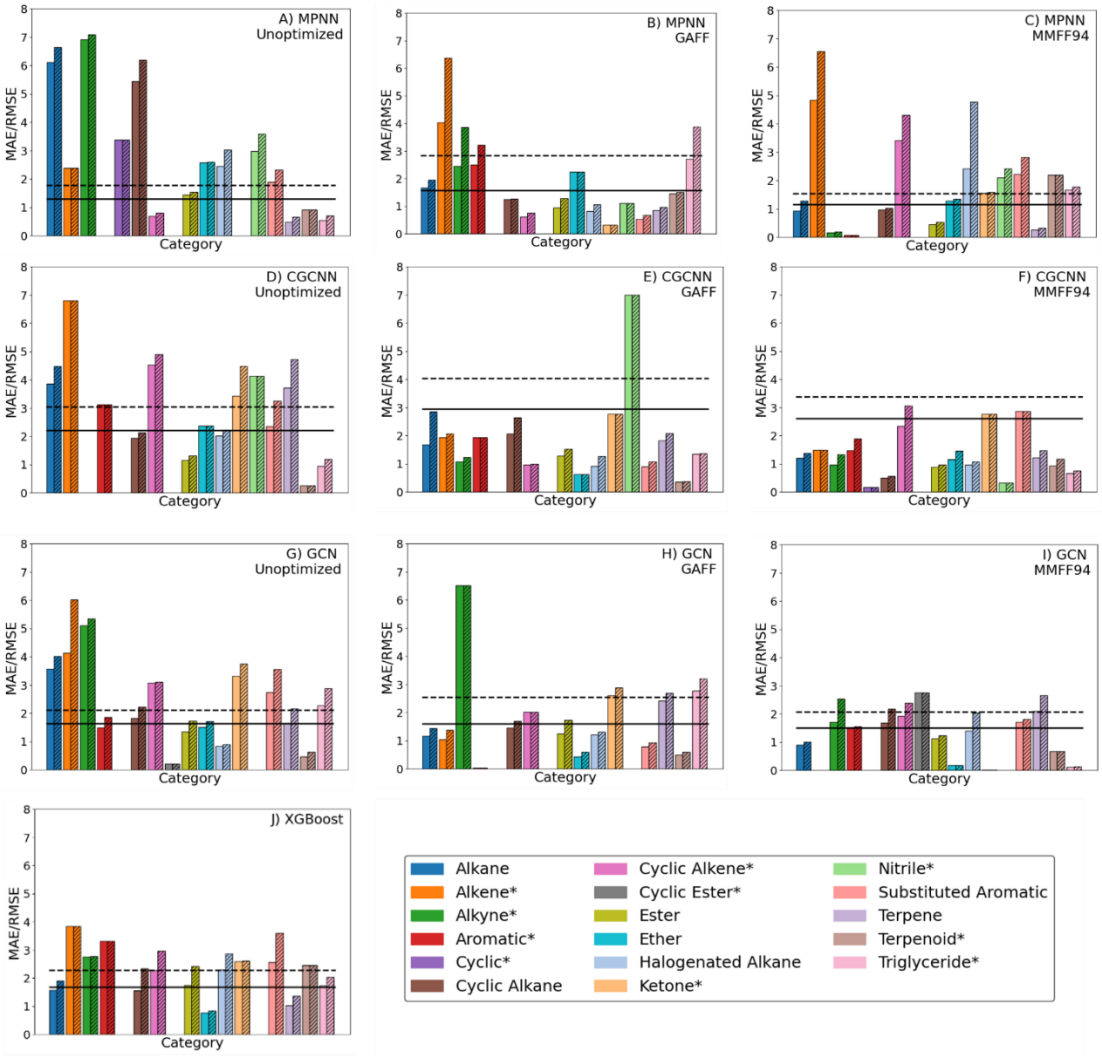

Figure 2.7. Testing MAE and RMSE for each category (Alkane, Alkene, Alkyne, Aromatic, Cyclic, Cyclic Alkane, Cyclic Alkene, Cyclic Ester, Ester, Ether, Halogenated Alkane, Ketone, Nitrile, Substituted Aromatic, Terpene, Terpenoid, sand Triglyceride), for the three GNNs with EACN data (A-I) and (J) XGBoost. With MAE shown in solid bars and RMSE shown in hashed bars. Categories which do not appear in the testing set are shown with no bar. The overall MAE is shown as a black solid horizontal bar and the overall RMSE is shown as a black solid horizontal bar. Functional groups with ten or less molecules in the dataset are shown with an * in the legend.

mixed. The EACN for linear alkanes equals the number of carbon atoms, so it is interesting that the models did not capture this definition well. The models trained with non-optimized geometry performed poorly, consistent with the overall results. MPNN performed poorly with alkenes; GCN, trained on geometry optimized with GAFF, performed poorly with alkynes; and XGBoost performed poorly with both alkanes and alkenes. The other models performed well on these oils. For ketones, both MPNN trained using geometry optimized with GAFF and GCN trained using geometry optimized with MMFF94 had almost zero error, and XGBoost also had a low error, whereas the other models performed poorly. All of the models performed moderately well with aromatics, cyclics, cyclic alkanes, cyclic alkenes, and cyclic esters. Finally, the models performed poorly on nitriles, except that CGCNN trained with geometry optimized with MMFF94 and MPNN trained with geometry optimized with GAFF had low error. The poor results could be attributed to the dataset's small number of nitriles.

Utilizing the F-statistic shown in equation 2.1 to evaluate the effects of functional groups on model performance. The models trained with non-optimized data have higher SSBs and SSWs than the other models for all bases in both the training and testing sets, and a higher F-Statistic than all other models except the CGCNN model trained with data optimized using MMFF94, which had a high F-Statistic on the training set. In general, models trained with data optimized with GAFF and MMFF94 had similar results with F-Statistics between ten and 12 for the training set, with the GCN model showing the best F-Statistic for the training set and between seven and nine for the testing set, except for

CGCNN trained using data optimized with MMFF94 for the training set. However, models trained on data optimized with GAFF have higher SSBs and SSWs on the training set, whereas the opposite holds for the testing set. Finally, XGBoost has the best SSB and SSW on the training set, with a negligibly lower F-Statistic (11.57 versus 11.58). For the testing set, the SSB and SSB are higher than those of several GNNs; however, the F-Statistic is the best among all the models. Overall, models trained on optimized data outperform those trained on non-optimized data, with XGBoost achieving the best performance.

The work in this chapter predicted the EACN of oil molecules, both without and with geometry optimization, using different force fields (GAFF and MMFF94), and employed molecular descriptors to train three GNNs and an XGBoost ML model. To measure model performance, three key metrics were used: R^2, measuring the goodness of fit of the models; MAE, measuring the mean error of the predictions; and RMSE, measuring the precision of the predictions, which are displayed in **Table 2.1** and in bar plots and regression plots in **Figure 2.7**. Finally, this chapter examined how each model performed in predicting EACN across different organic functional groups.

Conclusion

The ML models studied can accurately predict the EACN from molecular data represented as molecular graphs. Training a GNN, or using molecular descriptors, is used to train an XGBoost model without needing large databases or the computational expense of a molecular simulation. The model predictions are significantly faster than experimental

and computational methods, providing an accurate, high-throughput screening method for determining the EACN of molecules. Models trained on geometry-optimized data performed better than those trained on non-optimized geometry, at the expense of longer runtimes. The XGBoost model performed well at a fraction of the runtime, outperforming all GNNs trained on non-optimized geometry and most GNNs trained on optimized geometries. Additionally, XGBoost's fast runtime and performance make it an optimal model, rendering training GNNs on non-optimized geometry redundant, as XGBoost outperforms these models in both performance and speed. The MPNN model overfitted to the training set, with significantly better results on the training set compared to the testing set; the other models did not have this issue and required less time to train. The overfitting could be due to the small dataset combined with the large size of the model (1.5×10^6 parameters). XGBoost had the most precise predictions, with narrower confidence and prediction intervals and a lower RMSE than the GNNs.

These HLD parameters guide μE formulations by quantifying how formulation variables govern phase behavior toward the optimal bicontinuous Type III/IV systems required for RFB electrolytes. The EACN term models oil hydrophobicity, requiring purification of complex redox-active oils to >98% for direct measurement. Since one objective of using the empirical HLD equation is to determine compositions that yield μE phase behavior, the results provide a fairly simple tool for estimating HLD parameters to guide μE formulations. The most difficult parameters to obtain for the HDL equation are

the EACN and Cc terms used to model μE systems to find type III and type IV μEs at HLD ≈ 0.

CHAPTER 3:

PREDICTING CRITICAL MICELLE CONCENTRATION AND THE EFFECTS OF TEMPERATURE ON MICELLIZATION USING MACHINE LEARNING

Abstract

The work in this chapter uses ML models to predict the CMC of surfactants across different temperatures and headgroup structures, analyzing how temperature and headgroup structure influence model predictions. Five ML models were trained using a dataset of 1,110 known CMC values of 547 surfactants at different temperatures. The ML models consist of three GNNs (CGCNN, GCN, and MPNN), and two traditional ML models (XGBoost and GP models) were used, providing comparisons of both model structure and data representations for predicting CMC. The work in this chapter further analyzes the effectiveness of the ML models across three categories of surfactants: cationic, anionic, and nonionic. The results demonstrate the potential of ML for predicting temperature-dependent CMC across headgroup chemistries. The ML models enable high-throughput, accurate screening of surfactants with temperature analysis.

Introduction

The CMC is determined by the standard free energy of micellization; it represents the concentration at which surfactant aggregation becomes thermodynamically favorable, not an inherent property of the surfactant itself. However, this concentration affects various other properties, such as conductivity,[80,89] surface tension,[43,77,80,89] osmotic pressure,[89] absorption,[89] equivalent conductivity,[304] and interfacial tension.[88] The CMC is the surfactant concentration in water at which the transition from a monomeric dispersion to efficient packing (geometric arrangement of the surfactant) of the micelle occurs. The work

in this chapter further analyzes the effectiveness of the ML models across three categories of surfactants: cationic, anionic, and nonionic. Knowledge of the CMC is important for developing surfactant-based systems,[77,78,86,88,90–92,305] with applications in various industries, including pharmaceuticals,[81] food science,[37] and materials engineering.[306] Micelles are nanoscopic, spherical aggregates of surfactant molecules. Surfactant molecules have a hydrophilic headgroup and hydrophobic tails. In an aqueous solution, the tails cluster in the micelle core, shielded from water, while the polar headgroups face the surrounding aqueous environment. The surfactant polarity allows micellar systems to disperse nonpolar, oily components in an aqueous medium. External factors affecting CMC include temperature, pH, and the presence of other chemical species.[87,307,308]

Various experimental methods are available to determine CMC, including resonance Rayleigh scattering,[80] surface tension measurements,[77,309] conductivity measurements,[310] fluorescence changes,[84] turbidity,[91] and nuclear magnetic resonance spectroscopy.[90] These techniques work by detecting changes in the aforementioned properties relative to changes in the surfactant concentration in the solution. Computational approaches include molecular simulation software, such as COSMO-RS, which provides CMC estimates.[88,92] Computational models provide only approximate results because they simplify molecular interaction parameters and are primarily designed for neutral systems, whereas only 37.4% of the surfactants in the dataset used in this chapter are neutral.[311,312]

To address the throughput limitations of experimental methods and the lack of generalizability of simulation software in determining CMC at high-throughput, ML

techniques are used. Compared to traditional approaches, trained ML models offer significant advantages in speed, throughput, ease of use, and generalizability. Rooted in statistical principles, ML models predict target properties from structured input data and encompass a range of architectures, including NNs capable of capturing complex, nonlinear relationships through hierarchical feature learning; tree-based models such as XGBoost; and Bayesian inference methods, such as GPs with built-in uncertainty quantification.

Among NN architectures, GNNs have shown promise for chemical applications, as they represent molecular structures as graphs rather than fixed-length descriptor vectors, allowing for a more natural encoding of chemical structure. XGBoost, by contrast, leverages gradient-based optimization to improve decision trees through GTB iteratively, minimizing statistical loss functions to achieve high predictive accuracy. Meanwhile, GPs provide a probabilistic framework for prediction, offering both mean estimates and uncertainty quantification via kernel-based Bayesian inference.

Several notable studies have explored the application of ML to predict the CMC in various conditions and types of surfactants. The work in Brozos et al. utilized GNNs to predict CMC, enhanced with ensemble and transfer learning, using data with various tail groups measured using different methods.[93] Chen et al. use different model structures, namely tree-based ensemble algorithms, using a similar, although larger dataset to Brozos et al. [85,93] Rather than using graphs to train a GNN, Chen et al. generated 208 descriptors using RDKit, which was reduced to 19 using RFECV, finally conducting a feature analysis

using correlation plots and SHAP analysis, which verified the importance of temperature for nonionic surfactants. The work in Seddon et al. used a hybrid approach, combining the ML model with the Szyszkowski equation to verify the results.[92] An early study presented in Huibers, et al. leveraged QSPR to predict CMC of nonionic surfactants, generating a simple model with only four coefficients that could predict a subset of CMCs using three descriptors, comparable results in table 3 within Huibers, et al.[86] Finally, Nivón-Ramírez, et al. utilized COSMO-RS to predict the CMC of SDS modeled with water and verified using isothermal titration calorimetry experiments, which showed similar results to the simulated CMCs.

Each of these studies leveraged temperature differently. In Brozos et al., the temperature ranges were limited (20-28°C) and were not used as part of the input of the models, as a temperature change within this narrow range would not have a meaningful effect on CMC.[93] Chen et al. used a wider temperature range (20-50°C), considering temperature changes within this range as a factor, although not sufficient to cause a significant change in CMC. While the limited effect of temperature is confirmed using SHAP values in Figures 7b and 7c of Chen et al., one aspect not considered is the effect of surfactants that degrade at temperatures below 50°C.[85] Himanen et al. considered temperature the only non-molecular feature included, similar to the work in this chapter, highlighting its importance, whereas Huibers et al. did not consider temperature a predictive factor. Finally, the study by Nivón-Ramírez et al. used COSMO-RS at various temperatures (15-35°C), providing insights into the effects of temperature on

micellization.[88] None of these studies explores the effects temperature has on the outputs made by a model in a generalized manner, either considering CMC to be mostly invariant to temperature or only using a small subset of surfactants that don't adequately capture the entire relationship between CMC and temperature, and how this correlation affects ML models.

The data used in the work for this chapter uses 1,110 known CMC values collected from the National Bureau of Standards (now known as NIST) over three surfactant headgroups: cationic, anionic, and nonionic, with a broad temperature range of 0°C-90°C.[79] The three GNNs incorporate molecular structure and temperature directly into the graph representation as a surrogate node, whereas the XGBoost and GP models rely on RDKit-derived molecular descriptors that are augmented with temperature, this pipeline is summarized in **Figure 3.1** By comparing these distinct model classes and data representations under a common framework, this chapter can assess the trade-offs between accuracy, runtime, and temperature sensitivity, providing practical guidance for model selection in CMC prediction and high-throughput surfactant screening.

Methods

A NN is an ML model composed of interconnected layers that excel at capturing non-linear relationships in complex data. GNNs extend this architecture to process graph-structured data, such as molecules, unlike traditional NNs, which are designed for fixed-dimensional inputs.[52,167–169,222–230] Due to the heterogeneous nature of graph data, GNNs utilize specialized graph

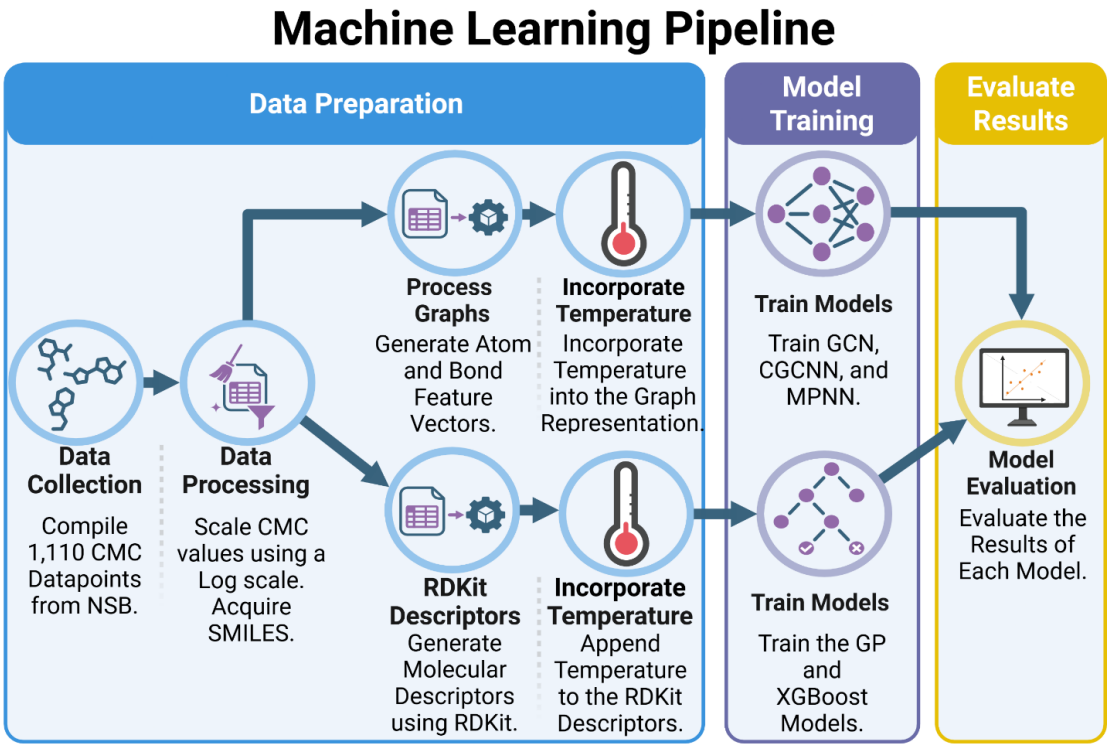

Figure 3.1. ML workflow for GNNs (Top) and XGBoost/Gaussian process (Bottom). Molecular structure and corresponding SMILES representation (inputs); data processing and model training for GNNs consists of SMILES to XYZ plots with geometry optimization followed by MatDeepLearn which generate adjacency matrices to atom and bond feature vectors with an additional feature vector for GNN model training. Data processing and model training for XGBoost and Gaussian Process consists of SMILES to molecular descriptors, followed by feature elimination and model training both GNNs and XGBoost/Gaussian Process output regression plots and evaluation metrics; MAE, RMSE, and R^2. Created in BioRender. Furth, N. (2026) https://BioRender.com/ayu9zgd

convolutional and pooling layers at the beginning of the network, iteratively reducing the graphs into a fixed-dimensional representation while preserving key structural and feature information. Molecular graph data uses graphs where nodes represent atoms and edges represent bonds.

This chapter examines three GNN architectures selected for their effectiveness in predicting chemical properties from molecular graph data: CGCNN, GCN, and MPNN.[168,169,222,226,228,230,238] The models were implemented using MatDeepLearn, a benchmarking framework designed for materials science applications that leverages state-of-the-art GNNs.[224] The search space covered multiple aspects of the GNN architectures, including dense layer dimensions, pooling layer types (mean, max, add, set-2-set), and the number of convolutional and dense layers. Given the large hyperparameter space of the GNNs, an exhaustive search was impractical. Instead, a randomized search was conducted, selecting a limited number of hyperparameter combinations to approximate the optimal configuration of each model without adding significant computational burden. The final optimal hyperparameters chosen for each GNN are summarized in **Table 3.1**.

XGBoost is an ensemble learning algorithm based on the GTB algorithm, which iteratively improves predictions by adding decision trees sequentially.[180,313,314] XGBoost refines predictions by sequentially training new trees to reduce errors from the previous one. A key characteristic of GTB is that it optimizes the model by applying gradient-based methods to minimize a specified loss function, unlike standard tree-based models. Instead of training multiple models in parallel like conventional ensemble methods, GTB starts

Table 3.1. Selected Hyperparameters for the three GNNs; MPNN, CGCNN, and GCN trained using data with geometries optimized using GAFF.

Model	Dense Layer Dimensions			Graph Convolution Count	Dense Layer Count	Pooling Method	Number of Model Parameters
MPNN	80	150	50	4	3	Max Pool	1,564,921
CGCNN	100	150	-	4	3	Max Pool	274,001
GCN	170	40	-	1	1	Set-2-Set	56.351

with a single model and progressively expands the ensemble to minimize errors. XGBoost enhances traditional GTB by introducing regularization techniques, such as L1 and L2, to prevent overfitting by controlling disproportionately large coefficient values through the regularization parameter γ. However, γ=0 was determined to be optimal, so γ was not used. Additionally, XGBoost requires no imputation for datasets with missing values, allowing more data to be utilized during training. Another advantage of XGBoost is its support for parallelization, enabling faster training and improved computational efficiency. The optimal hyperparameters identified via grid-search CV for XGBoost are presented in **Table 3.2**.

A GP is a Bayesian-based statistical method that models functions using a Gaussian distribution as a prior.[148,160,248] A GP defines a prior distribution function $f(\cdot)$ such that $f(\cdot): x \rightarrow y$, and explicitly quantifies uncertainty, making it well-suited for cases where uncertainty metrics are essential. A GP model is mathematically represented as shown in equations 3.1 and 3.2:

$$f(\cdot) \sim GP_M\big(h(\cdot), C(\cdot,\cdot)\big) \tag{3.1}$$

$$y_n \mid f(\cdot) \sim Normal_M(f(x_n), v_n), v_n = \epsilon_n^2 \tag{3.2}$$

Table 3.2. Selected Hyperparameters for XGBoost trained using molecular descriptors.

Optimization Method	Model	Max Depth	Max Estimators	Regularization parameter-γ
Molecular Descriptors	XGBoost	2	17	0

In equations 3.1 and 3.2, M is the dimensionality of the data, $h(\cdot)$ is the mean function of the Gaussian prior, typically set to zero. The covariance kernel, $C(\cdot,\cdot)$, captures relationships between datapoints.[160] The model is trained in a manner where the observed outputs are expressed as $y_n = f(x_n) + \varepsilon_n$. The term ϵ_n represents the standard deviation of each observation, modeling inherent noise, and $n \in N$ is the number of datapoints.

The GP covariance kernel determines the predictive behavior of the GP model. Commonly used covariance kernels include quadratic, exponential, rational quadratic, white noise, and combinations of multiple kernels. For the GP used in this chapter, the radial basis function kernel is employed. The kernel function includes a shape parameter λ which controls the variance of the predictions, while the lengthscale parameter l determines the smoothness of the fitted function. Both of these parameters influence the model's generalization ability, providing better control than NNs. Finally, an α term represents a noise parameter. Once training data (x_n, y_n) is observed, the prior distribution is updated to form a posterior distribution, which is also Gaussian, with a variance that quantifies the uncertainty metric. The posterior Gaussian enables predictions for unknown datapoints $(x*, y*)$ conditioned on the observed data and the covariance kernel. The selected hyperparameters for the GP model are shown in **Table 3.3**.

The dataset contains 547 unique surfactant molecules with known CMC values across various temperatures, yielding 1,110 data points.[79] Of these 1,110 data points, four

Table 3.3. Selected Hyperparameters for the Gaussian Process trained using molecular descriptors.

Optimization Method	Model	Shape parameter-λ	Length scale parameter-l	Noise parameter-α
Molecular Descriptors	Gaussian Process	2	0.1	0.8

(0.36%) were excluded from GNN training due to the SMILES string for dodecyl N betaine hydrochloride failing graph generation. The graph generation failure is likely due to the SMILES-to-xyz conversion used by Open Babel, which produces empty values for certain complex molecules. The CMC values in the dataset range from ~10^{-6} to 10 M. Given the wide range of magnitudes of the CMC values shown in **Figure 3.2**, the CMC is scaled using a log-scale $[-log_{10}(M)]$ whose distribution is shown in **Figure 2B**, the negative term doesn't change the scale of the units or affect the predictions, only improving the visualization. The log transformation helps prevent ML models from disproportionately fitting larger values by adjusting their scale, making optimization for mean error more balanced by reducing sensitivity to extreme values. The original CMC distribution shown in **Figure 3.2** is heavily skewed towards zero, with values mostly ranging from 10^{-4} to 10^{-1} M. After the log transformation, the CMC distribution more closely resembles a normal distribution. The temperature of the molecules ranges from 0 to 90°C, as shown in **Figure 3.2**. The temperature distribution has a mean of 25°C, which is expected, since many experiments are conducted at room temperature. The temperature distribution approximates a sparse non-symmetric normal distribution skewed towards the higher range; this skewedness is because no measurements were recorded below 0°C, which is impractical due to effects from water on micellization, whereas the log-transformed CMC distribution is more symmetrical. The data used to train ML models consists of two components: the target (CMC) to be predicted and the input features (graphs, RDKit

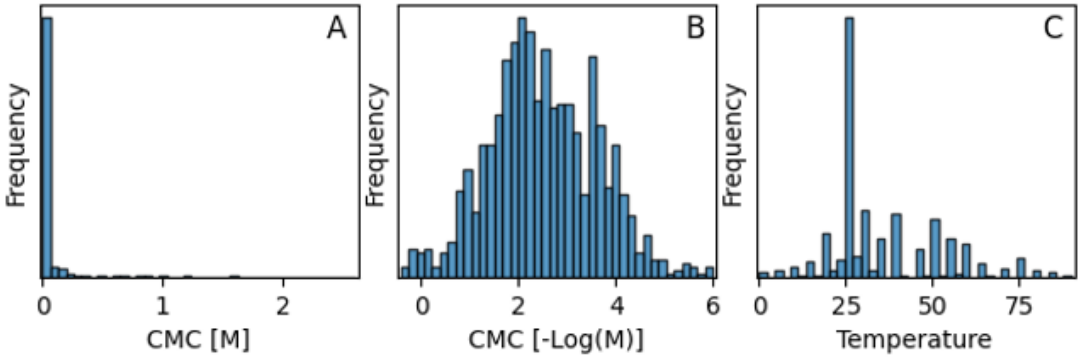

Figure 3.2. Histogram of A) CMC in molarity (M), B) CMC in -Log(M), and C) temperature distribution in Celsius (°C).

descriptors, and temperature) used to predict the target. The ML pipeline used in this work is shown in **Figure 3.1**. The GNNs (upper path) used the MatDeepLearn framework.[224]

In MatDeepLearn, first, the molecules are represented as SMILES strings. Then the SMILES are used to generate atomic structures in ASE-compatible formats (i.e., xyz format). These atomic structures are then converted to graph objects using sparse bond connection lists, containing the atom indices at both ends of each bond, having a size of $2 \times M$, where M is the number of bonds. The atom feature vectors are unique for each atom in each molecule, containing the one-hot encoded atomic number and the one-hot encoded number of neighbors (other atoms within the predefined max cutoff distance in Å). The total number of neighbors is equal to or less than the predefined max neighbors. To incorporate temperature as a non-structural feature into the graph data, the molecular graph is extended with a surrogate temperature node. Specifically, an extra element is appended to each atom feature vector and initialized to zero, and the corresponding entry for the designated surrogate node is set to the experimental temperature for that datapoint; while all other atoms remain zero in this dimension, this process is shown in **Figure 3.3**. Atom and bond feature vectors are then augmented with the temperature node at the newly generated index, enabling convolutional layers to propagate temperature information across the graph properly. The graph-native approach used in this chapter contrasts with a simpler descriptor concatenation in the XGBoost and GP models, which avoids the complexity of the graph-based data used by GNNs.

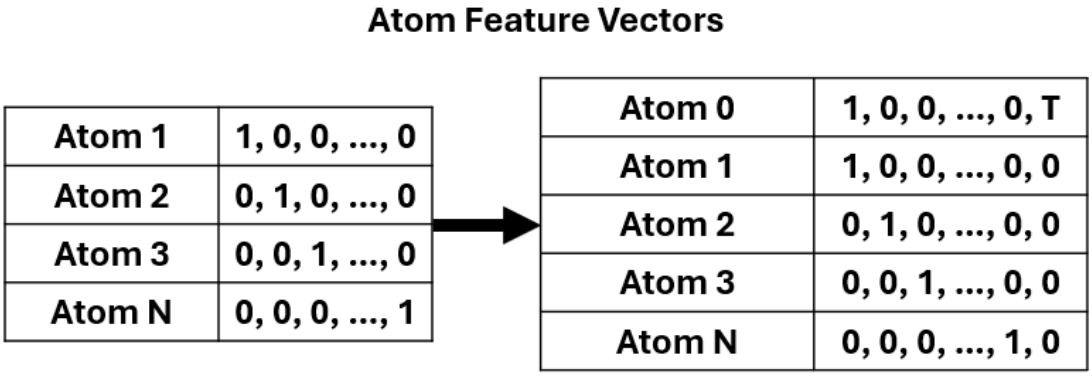

Figure 3.3. Illustration of appending a temperature surrogate node to graph data.

In contrast to the three GNNs, XGBoost and GP utilize 244 unique molecular descriptors generated using RDKit from the SMILES representations, with temperature added as an additional feature index.[215] The values of the molecular descriptors are then scaled to the range (-1, 1) to normalize the feature space. Scaling the data prevents numerical instability, ensuring fair feature weighting and accelerating model training. To train the XGBoost and GP models, not all 244 features from RDKit are necessary; RFECV is utilized to optimize feature selection by ranking each feature based on its correlation to the target variable (CMC) and its relationship to other features. The selection process involves iteratively training a surrogate model, starting with only the highest-ranked feature, and then progressively adding additional features one at a time in descending order of importance, training a new model each time. The iterations continue until the model has been trained with the entire feature set. Then the optimal number of features is selected using the evaluation metrics of the RFECV surrogate model. In **Figure 3.4**, the RFECV plot displays the mean test score as a function of the selected features, with the vertical dashed line indicating the optimal number of features (71) used to train the XGBoost and GP models.[215] Similarly, the optimal hyperparameters for the XGBoost and GP models are determined using grid search CV, by exhaustively evaluating every combination of hyperparameters and selecting the optimal set. The final hyperparameters for each model are shown in **Tables 3.1-3.3**.

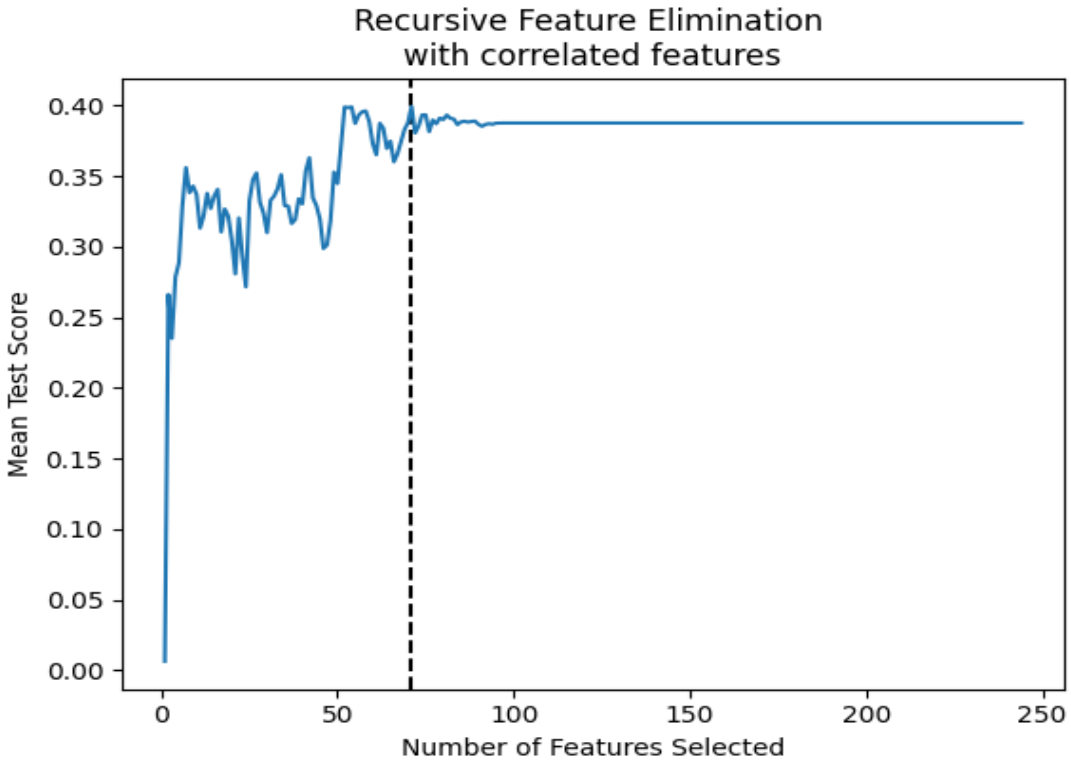

Figure 3.4. Recursive feature elimination with cross-validation graph for CMC. The vertical dotted line highlights the optimal number of features used in the XGBoost and GP models.

The study presented in this chapter trains and benchmarks five ML models: three GNNs (CGCNN, GCN, MPNN), XGBoost, and a GP, using a dataset of 1,110 CMC values from 547 surfactants spanning 0-90°C and three headgroup types (cationic, anionic, nonionic). Model performance was evaluated using MAE, root mean squared error (RMSE), and R^2-score across training/testing splits to assess accuracy-generalization tradeoffs. Additionally, temperature sensitivity and headgroup-specific predictions are analyzed to reveal how input encoding and architecture influence CMC forecasting for high-throughput surfactant screening.

Results and Discussion

To assess the performance of the models in predicting CMC, first, the runtime of each model is compared, then their training and testing performance is evaluated using three key metrics: MAE, RMSE, and R^2. The relationship between temperature and prediction error across models is examined to analyze how each model incorporates temperature as an input feature and how temperature affects the predictions made by the models. Finally, the model performance across three surfactant headgroup types: cationic, anionic, and nonionic was evaluated.

The runtime shown in **Table 3.4** is key for determining which ML methods provide the desired accuracy-speed tradeoff.[315] The execution time depends on multiple factors, including hardware and the computational complexity of operations, such as matrix calculations, partial derivatives, and convolutional operations. As expected, the largest GNN: MPNN had the longest runtime, three to seven times longer than CGCNN and GCN,

Table 3.4. Training runtime in seconds of MPNN, CGCNN, and GCN trained using geometries optimized with GAFF. XGBoost, and GP on data generated using molecular descriptors predicting CMC.

Model	Run Time (s)
MPNN	762
CGCNN	119
GCN	212
XGBoost	0.0140
Gaussian Process	2.28

respectively, due to its larger number of parameters. In contrast, XGBoost was the fastest model, completing training in approximately 14 milliseconds, making it over 8,500× faster than the fastest GNN. Finally, the GP model took just over two seconds, which, while slower than XGBoost, remains significantly faster than the three GNNs. Interestingly, CGCNN runs twice as fast as GCN (212 seconds vs 119 seconds) despite having five times as many parameters, due to the different convolutional operations used, whereas the slowest model, MPNN, completed training in just under 13 minutes. After training an ML model, evaluating new molecules requires significantly less time, as gradient descent and iterative parameter updates cease, resulting in fewer operations. These runtime comparisons do not account for accuracy or precision. Still, they highlight the substantial impact that model selection has on training time and the advantages of ML over traditional approaches.

All five models, CGCNN, GCN, MPNN, XGBoost, and GP, generalized well across the dataset, exhibiting only minor differences between training and testing metrics for MAE, RMSE, and R^2, as shown in **Figure 3.5**. The models generalized well, exhibiting minor differences in all three metrics between the training and testing sets. Among the five models, CGCNN achieved the best overall performance for both training and testing sets, as shown in **Table 3.5**. For the training set, MPNN and GCN performed similarly to each other but worse than CGCNN. However, MPNN showed slightly better test set results than CGCNN in RMSE (0.452 vs. 0.428) and R^2 (0.822 vs. 0.835), though CGCNN remained the top-performing model when considering all metrics combined. The GCN model was

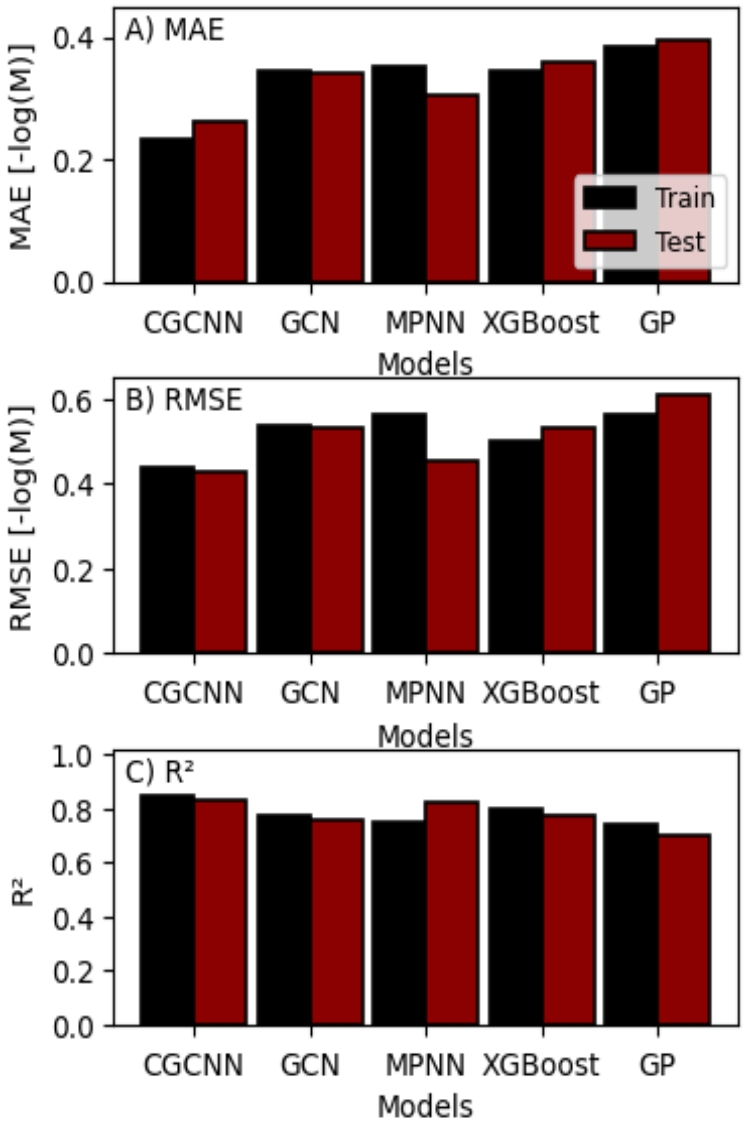

Figure 3.5. Training (black) and testing (red) results ((A) MAE, (B) RMSE, and (C) R^2) for the three GNNs with CMC data, CGCNN, GCN, and MPNN trained with data optimized using GAFF and molecular descriptors with XGBoost Gaussian Process.

Table 3.5. MAE, RMSE, and R2 for the training and testing datasets for each model evaluated (MPNN, CGCNN, GCN, XGBoost, and GP) trained using CMC data. Additionally, the MPNN, CGCNN, and GCN have been compared with different molecular geometries, graph data optimized using GAFF, and molecular descriptors used with XGBoost and GP.

Optimization Method	Model	Training			Testing		
		MAE	RMSE	R^2	MAE	RMSE	R^2
GAFF	MPNN	0.321	0.526	0.787	0.304	0.452	0.822
	CGCNN	0.204	0.393	0.877	0.261	0.428	0.835
	GCN	0.325	0.532	0.785	0.339	0.530	0.760
Molecular Descriptors	XGBoost	0.344	0.501	0.795	0.360	0.532	0.770
	GP	0.376	0.553	0.751	0.393	0.607	0.702

the worst of the three GNNs, with the testing set, with an R^2 of 0.760, MAE of 0.339, and RMSE of 0.530; the full numerical values for all five models are shown in **Table 3.5**. These results for the GNNs demonstrate that model size alone does not determine performance. The GCN (274,001 parameters) underperformed compared to the largest MPNN (1,564,921 parameters), while CGCNN (56,351 parameters), despite being the smallest, achieved the best overall results, with full model sizes detailed in **Table 3.1**.

Instead, the type of convolution operation, which controls the final graph representation, proves more influential than raw model size. The geometry-aware processing used by CGCNN, which incorporates structural details like interatomic distances into its convolutions, outperforms

The basic neighbor aggregation used by GCN and the sophisticated message-passing mechanism used by MPNN, which iteratively updates node features based on neighbor interactions. CGCNN and GCN exhibited slight overfitting; in contrast, MPNN generalized most effectively, surpassing its training performance across all test metrics. Finally, the XGBoost and GP models performed worse than the GNNs in every aspect, as shown in **Table 3.5**, with the exception of faster runtimes. Interestingly, XGBoost did not overfit, despite the absence of the regularization term γ, whereas GP slightly overfit to the training data, even after adjusting model parameters to control for overfitting. Chapter two demonstrated that the larger models had a higher degree of overfitting. However, the dataset used in this chapter is ~6× larger.[22,224] The larger dataset used for the work

presented in this chapter better leveraged the MPNN model, exhibiting no significant overfitting and performing competitively with the other models, demonstrating the relationship between the dataset size and model complexity. These results support the hypothesis presented in chapter two: that larger GNN architectures, such as MPNN, are more prone to overfitting when trained on smaller datasets.[22]

To examine why the GNNs performed better than the XGBoost and GP models, the MatDeepLearn encoding is related to CMC to expand interpretability. The bond lengths enable the GNNs to infer surfactant hydrophobic chain length. Longer linear alkyl chains favor efficient micelle core packing through enhanced van der Waals interactions, generally lowering CMC, whereas highly branched or irregular chains disrupt packing efficiency and tend to increase CMC.[316,317] Bond order can be inferred from interatomic distance distributions: unsaturated bonds introduce geometric constraints that reduce conformational flexibility and disrupt optimal tail packing, typically elevating CMC, while saturated bonds allow extended conformations that promote tighter core packing and lower CMC.[318,319] The atomic one-hot encodings capture compositional effects: dominance of C and H drives hydrophobic tail character and low CMC, whereas incorporation of heteroatoms increases headgroup polarity and hydration, enabling GNNs to learn structure-polarity-packing relationships in an end-to-end manner.[320]

RDKit descriptors (244 total) are too numerous to relate individually to CMC, but they span structural and physicochemical categories relevant to micellization. These include 2D topological indices (Wiener/Zagreb numbers, atom/bond/torsion counts),

electronic and charge-related properties (Gasteiger charges, eigenvalues), hydrophobicity and solubility metrics (logP, logS, PSA, SASA), flexibility measures (rotatable bonds, ring counts), and pharmacophoric features (H-bond donor/acceptor counts), along with 3D geometrical descriptors (asphericity, sphericity, inertia tensors, radius of gyration), steric/shape indices (PMR, PMI ratios), and spatial autocorrelations. Collectively, these features approximate factors governing surfactant packing: 2D logP and carbon-count-related descriptors capture hydrophobic driving force (e.g., lower CMC for C12-C16 chains), while 3D shape metrics such as asphericity reflect molecular elongation and compatibility with micelle core curvature. PSA/SASA-related descriptors quantify hydrophilic-hydrophobic balance relevant to amphiphile stability.

The GNNs outperformed RDKit descriptors combined with XGBoost because MatDeepLearn explicitly encodes molecular 3D geometry through interatomic distances and bond connectivity. Bond lengths directly reflect hydrophobic tail extension, enabling more precise inference of packing efficiency, whereas the 2D topological indices and rotatable bond counts serve as indirect proxies generated by RDKit. Interatomic distance patterns in MatDeepLearn preserve local geometric information that reveals branching or unsaturation-induced kinks associated with reduced packing efficiency and elevated CMC, while RDKit 3D descriptors, such as inertia tensors or asphericity, condense this information into global averages that may dilute subtle structural signals in tabular regression. Atomic one-hot encodings further allow composition–polarity tradeoffs (hydrocarbon tails vs. polar headgroups) to be learned directly from data, rather than

through fixed correlations such as logP or PSA. Nevertheless, models trained on RDKit descriptors still performed well because CMC strongly correlates with simple structural features, particularly hydrophobic chain length and overall hydrophobicity, which are effectively captured by 2D descriptors such as carbon counts and logP.

After evaluating the performance metrics of each model directly, the $-log10(M)$ transform is reversed to evaluate the results using molarity, although for visual interpretability, the axis uses a log scale. To visually represent the predictions of each model, the predicted values for each datapoint are plotted against their actual values. In **Figure 3.6**, the x-axis represents the actual target values, while the y-axis shows the predicted values. The solid red line represents each fit line of each model, the solid blue line represents the ideal fit line, the dashed grey lines mark the 95% prediction interval, and the shaded blue region represents the 95% confidence interval of the model. The training set (encompassing both the training and validation data for the GNNs) is shown as black markers, while the testing set is shown as red markers.

Among the models, CGCNN demonstrated the tightest prediction clustering and narrowest 95% confidence intervals around its regression fit line, reflecting high predictive precision. An outlier is defined here as a prediction whose error exceeds 3σ (three standard deviations), where the standard deviation of the errors is expected for ~0.3% of points under normality assumptions. CGCNN identified exactly 31 such outliers (0.28%), with its fit slope closest to the ideal $y = \hat{y}$ fit line. In contrast, MPNN (20 outliers) and GCN (19 outliers) had fewer outliers than expected, while XGBoost (15) and GP (16) had the fewest

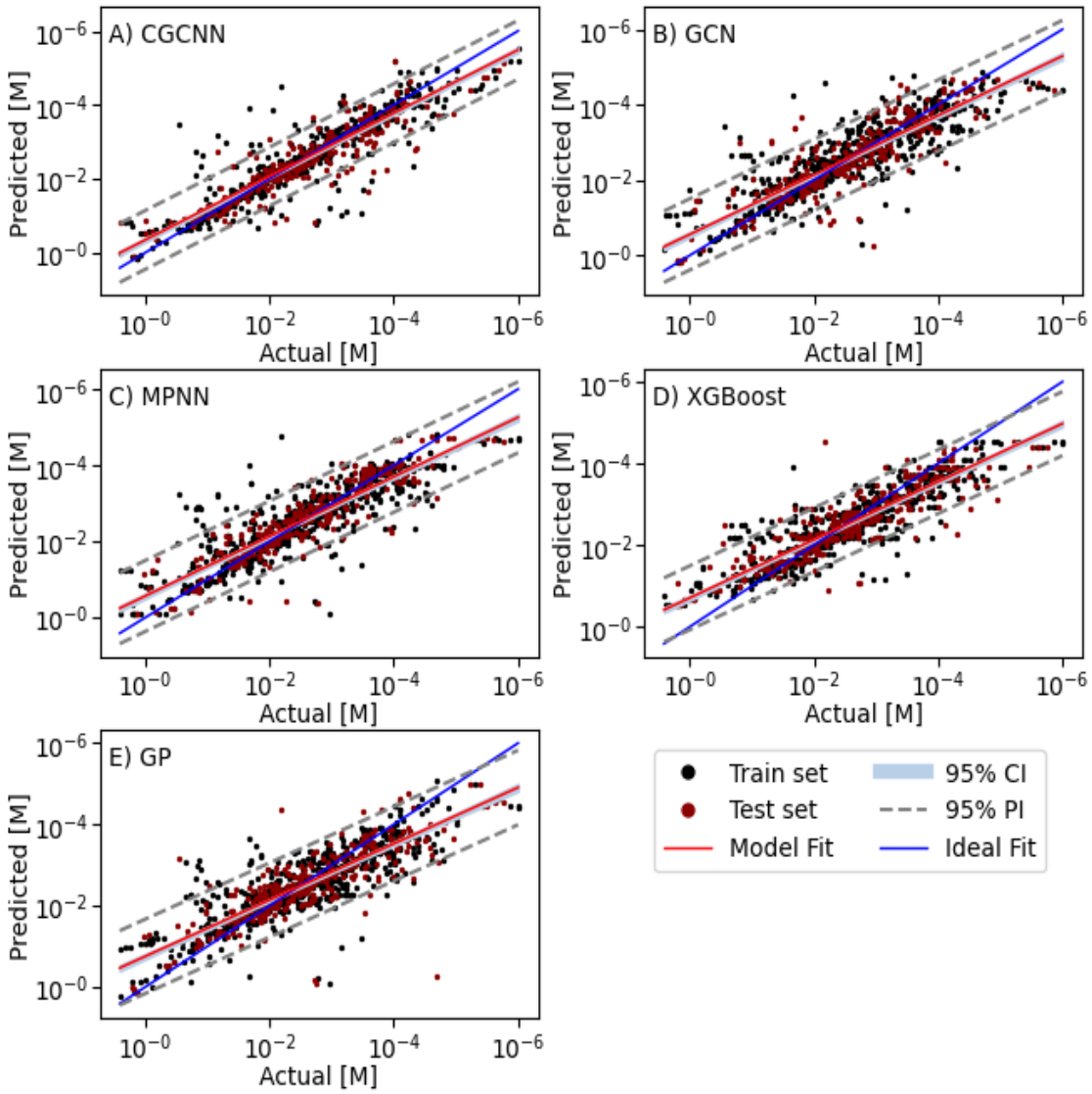

Figure 3.6. Regression plots for the three GNNs with CMC data, (A) CGCNN optimized using GAFF, (B) GCN optimized using GAFF, (C) MPNN, optimized using GAFF (D) molecular descriptors with XGBoost, and (E) molecular descriptors with Gaussian Process with the fit line of the model shown in black, the 95% confidence interval shown in blue, the 95% prediction limit shown as a dashed grey line, and the ideal fit as a blue line. Training data is shown in black, and testing data is shown in red. Horizontal bands are shown with a black arrow.

outliers. A counterintuitive pattern arises because poorer-performing models inflate σ through broader error distributions, raising the outlier threshold (3σ), masking their systematic errors within this inflated variability. Better models like CGCNN, with tighter error distributions and therefore lower σ, apply a stricter threshold.

Finally, the predictions made by XGBoost exhibit discrete horizontal bands in **Figure 3.6**, as demonstrated with a black circle and arrow. As tree splits prioritize high-variance molecular descriptors (e.g., BertzCT complexity, MolWt) over temperature, creating quantized outputs insensitive to small temperature variations within the dataset range (0-90°C). While prior SHAP analysis conducted by Chen et al. confirmed that the correlation of temperature to CMC is low, but not zero.[85] The runtime of XGBoost is ∼8,500× faster training trades subtle temperature dependence for efficiency, highlighting the accuracy-speed tradeoff offered by ML. For temperature-critical applications exceeding this range, XGBoost supports manual feature weighting and custom objective functions, boosting temperature importance in a physics-informed manner (akin to PINNs). Despite deliberately excluding temperature, XGBoost still outperformed the GP model across all metrics and was similar to GCN, demonstrating that overall, feature selection outweighs forcing in a low-value variable like temperature.

Evaluation metrics such as MAE, RMSE, and R^2 provide valuable insights into model performance. However, these metrics are sensitive to outliers and may not fully capture the distribution of prediction errors. Thus, examining the distribution of errors shown in **Figure 3.7**, in conjunction with MAE, RMSE, and R^2, and the regression plots

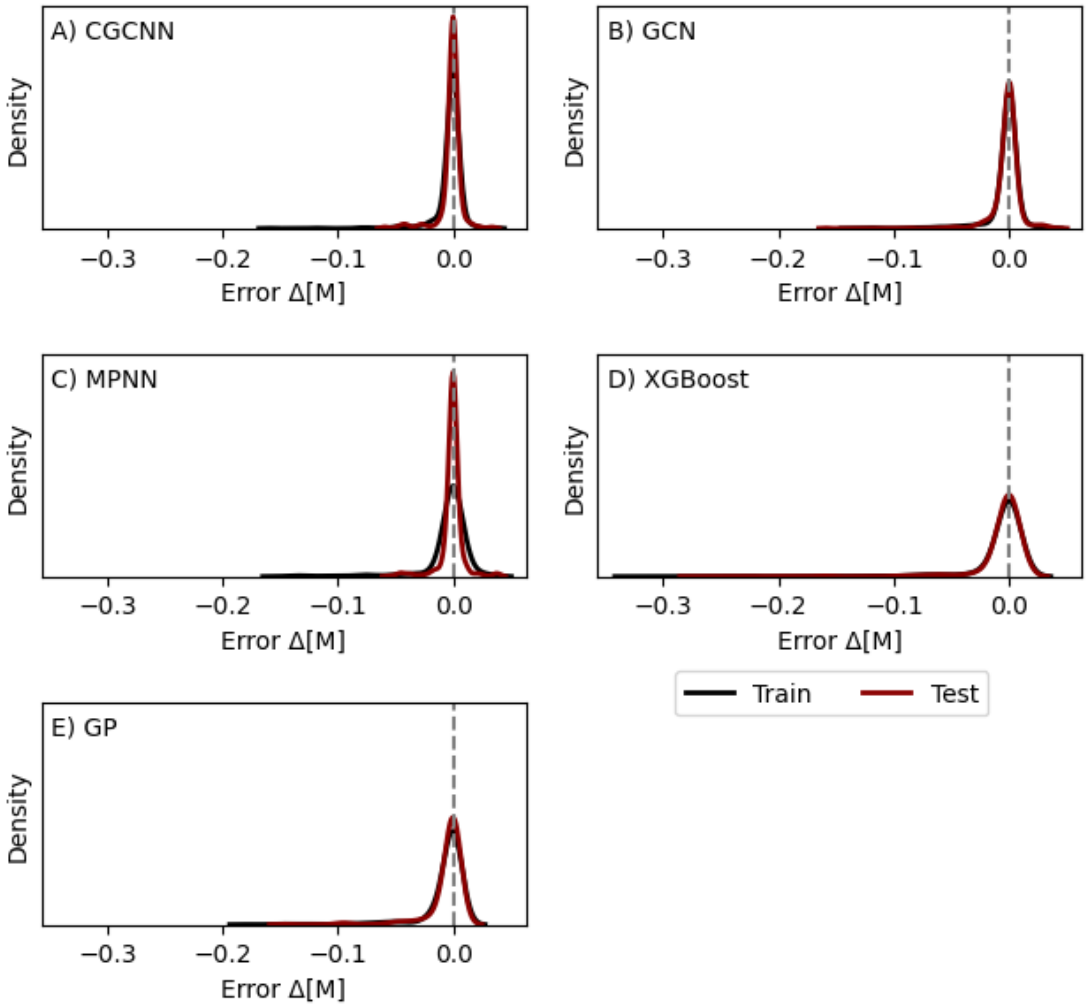

Figure 3.7. Distribution of absolute errors for each model evaluated (CGCNN, GCN, MPNN, XGBoost, and Gaussian Process) scaled to the same axis. (A) CGCNN optimized using GAFF, (B) GCN optimized using GAFF, (C) MPNN optimized using GAFF, (D) molecular descriptors with XGBoost, and (E) molecular descriptors with Gaussian Process, with the training set shown in black and the testing set shown in red.

in **Figure 3.6**, provideing a more comprehensive view of model performance. The error distributions for all five models exhibit approximately normal distributions centered at zero, with the 5% largest outlier errors removed to constrain the scale of the plots. CGCNN stands out with the narrowest distribution, having the densest peak at zero, and the tightest spread on training and testing data. MPNN is the most similar to CGCNN, especially with the testing set. The GCN model has a wider error distribution and more errors in the high negative range. The XGBoost and GP models show more negative asymmetry, meaning they tend to underpredict solubility more often than over, a conservative tilt visible in longer left tails. The three GNNs overall cluster more tightly than XGBoost or GP, consistent with their superior metrics, and the all-models aggregate in **Figure 3.7** confirms a negatively skewed distribution peaking at zero with tails to roughly -0.225M. **Figure 3.7** confirms that the ML models demonstrate suitable predictive performance for reliable surfactant screening across molecular structures and temperatures, with GNN-based models providing the most precise and balanced predictions.

Model performance compared to temperature as an input feature was evaluated as shown in **Figure 3.8**, providing insights into temperature generalization across all five models. **Figure 3.8** uses black dots for the training set and red dots for the test set. The plots show strikingly similar distributions for both train and test sets. The similarity between the training and testing sets indicates minimal overfitting with respect to temperature, as test errors mirror training errors closely at every temperature point. In denser regions around 20°C-30°C, both sets exhibit spikes in MAE, but these are following

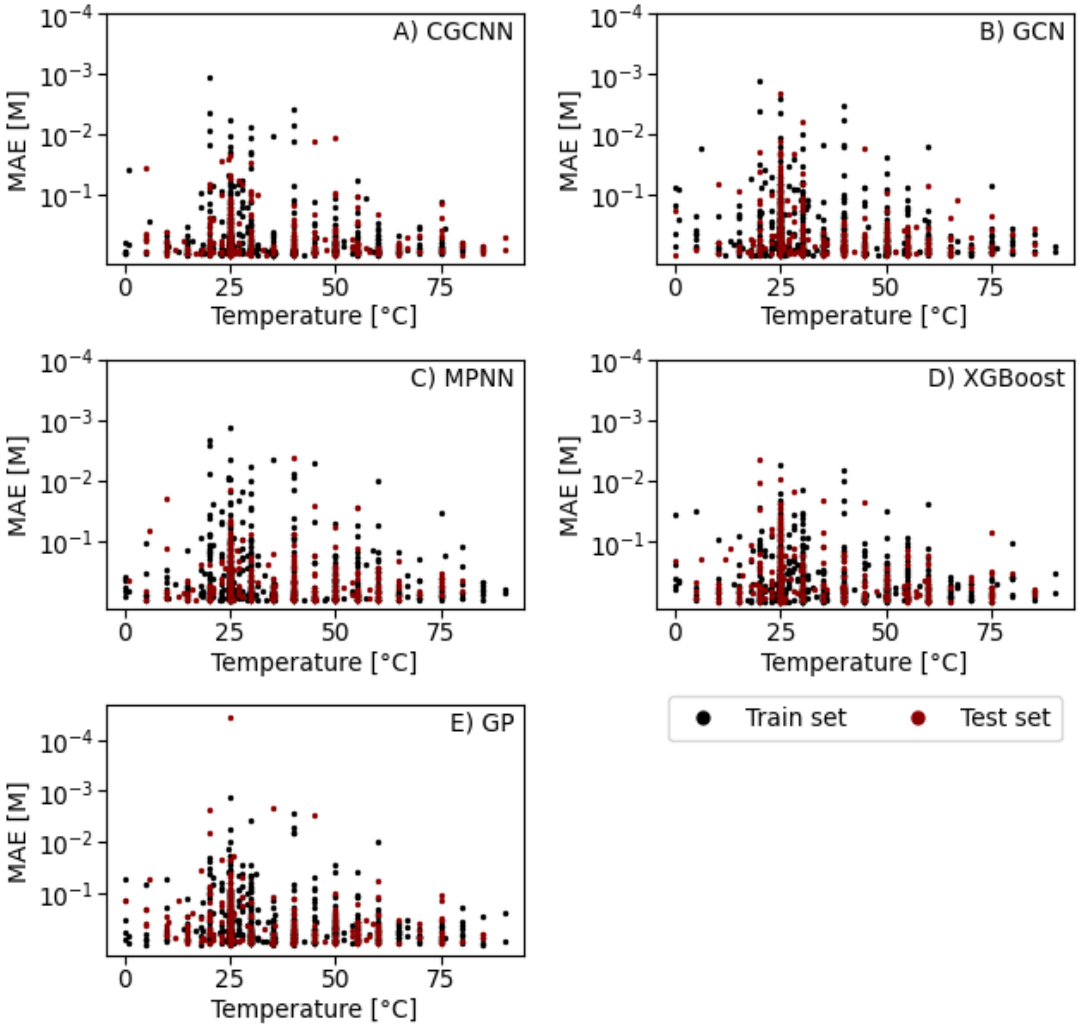

Figure 3.8. Error versus Temperature plots for each model evaluated (CGCNN, GCN, MPNN, XGBoost, and Gaussian Process). (A) CGCNN optimized using GAFF, (B) GCN optimized using GAFF, (C) MPNN, optimized using GAFF (D) molecular descriptors with XGBoost, and (E) molecular descriptors with Gaussian Process.

the temperature histogram from **Figure 3.2**, with peaks at 20-30°C matching the highest data density there. High density often amplifies apparent errors because models encounter more samples, making inconsistencies more visible statistically, even if per-sample accuracy remains consistent. Additionally, because XGBoost did not utilize temperature, it can be used as a temperature-invariant baseline to compare the other models to. Meaning, the incorporation of temperature as a surrogate node in the GNNs had no negative impact on model performance. Comparing the GNNs and GP to XGBoost, there is no significant difference in distribution, although there appear to be more outliers. Such temperature-dependent analysis is crucial in chemical informatics because CMC varies nonlinearly with temperature, demanding models that extrapolate reliably for untested conditions across anionic, cationic, and nonionic surfactants. Demonstrating data-driven errors over model flaws validates these GNNs and baselines for real-world deployment, where predictions guide high-throughput screening without retraining. It also highlights the need for targeted data augmentation in low-density areas (e.g., extremes near 0°C or 90°C), improving overall utility in predictive pipelines for surfactant design.

The surfactant dataset is distributed across three chemically distinct headgroups: cationic (n=224: 20.2%), anionic (n=471: 42.4%), and nonionic (n=415: 37.4%). Different headgroups create a class imbalance that can affect model generalization; the distribution is shown in **Figure 3.9**. **Tables 3.6** and **3.7** provide training and testing metrics (MAE, RMSE, R^2) for all five models across these three headgroups, whereas **Figure 3.10**, provides a visualization of the results. Across all surfactant headgroup categories, CGCNN

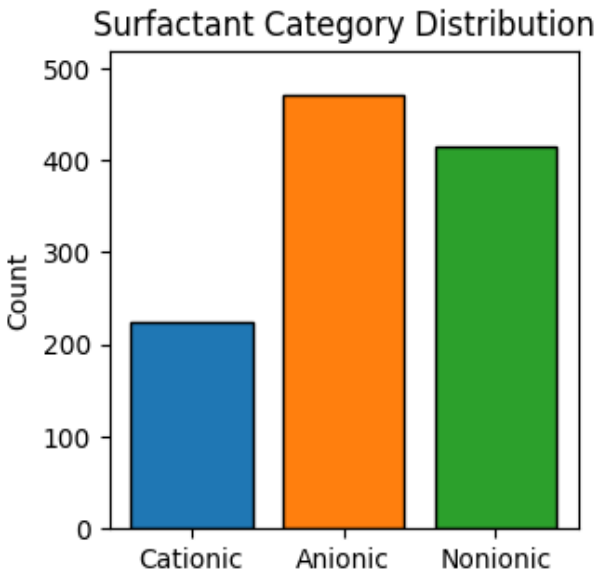

Figure 3.9. Distribution of headgroup classes: Cationic data (Blue), Anionic data (Orange), and Nonionic data (Green).

Table 3.6. MAE, RMSE, R^2 for the training dataset for each model evaluated (MPNN, CGCNN, GCN, XGBoost, and GP) trained using the full CMC dataset. Evaluated on three groups: cationic, anionic, and nonionic.

Model	Cationic			Anionic			Nonionic		
	MAE	RMSE	R^2	MAE	RMSE	R^2	MAE	RMSE	R^2
CGCNN	0.252	0.519	0.571	0.175	0.361	0.852	0.284	0.463	0.884
GCN	0.379	0.617	0.474	0.271	0.448	0.757	0.405	0.583	0.816
MPNN	0.401	0.654	0.412	0.328	0.528	0.675	0.351	0.544	0.838
XGB	0.415	0.606	0.384	0.284	0.421	0.785	0.374	0.523	0.849
GP	0.435	0.659	0.272	0.311	0.465	0.737	0.438	0.607	0.797

Table 3.7. MAE, RMSE, R^2 for the testing dataset for each model evaluated (MPNN, CGCNN, GCN, XGBoost, and GP) trained using the full CMC dataset. Evaluated on three groups; cationic, anionic, and nonionic.

Model	Cationic			Anionic			Nonionic		
	MAE	RMSE	R^2	MAE	RMSE	R^2	MAE	RMSE	R^2
CGCNN	0.333	0.542	0.597	0.230	0.406	0.775	0.260	0.381	0.908
GCN	0.332	0.529	0.491	0.248	0.411	0.804	0.441	0.634	0.750
MPNN	0.301	0.421	0.659	0.308	0.485	0.701	0.302	0.421	0.887
XGB	0.380	0.562	0.619	0.318	0.468	0.745	0.397	0.582	0.798
GP	0.361	0.536	0.654	0.331	0.524	0.681	0.484	0.721	0.690

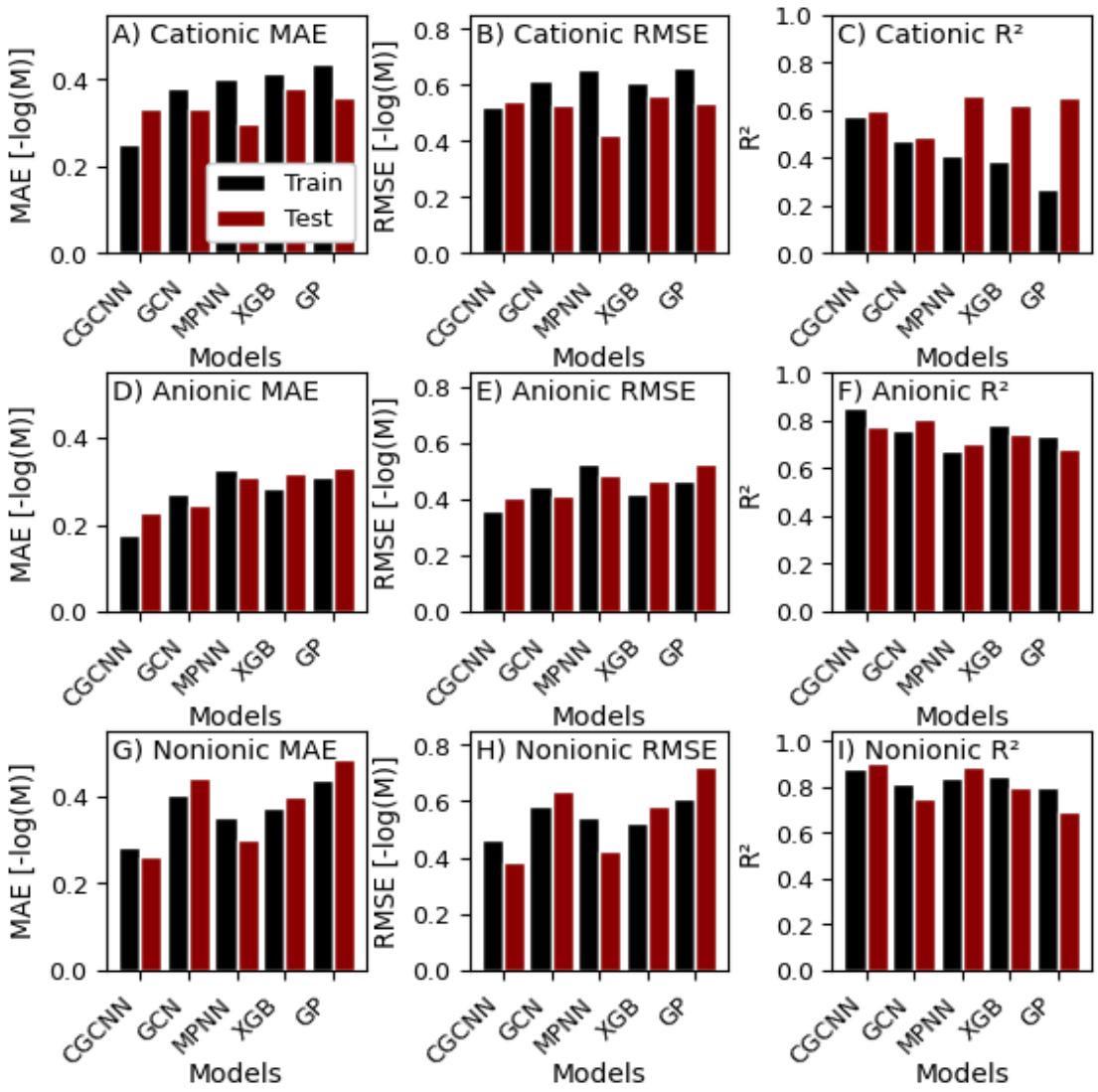

Figure 3.10. Training (black) and testing (red) results for the three GNNs with CMC data, CGCNN, GCN, and MPNN trained with data optimized using GAFF and molecular descriptors with XGBoost Gaussian Process with each category (cationic, anionic, and nonionic). (A) MAE for cationic, (B) RMSE for cationic, (C) R^2 for cationic, (D) MAE for anionic, (E) RMSE for anionic, (F) R^2 for anionic, (G) MAE for nonionic, (H) RMSE for nonionic, and (I) R^2 for nonionic.

consistently achieved the strongest performance and generalization, particularly for anionic and nonionic surfactants. XGBoost had comparable stability between training and testing sets. In contrast, the GCN, MPNN, and GP models underperformed relative to CGCNN, and finally, XGBoost showed higher errors and weaker generalization.

Model performance was best for the anionic surfactants due to their higher fraction of the data (n = 471, 42.4 % of the dataset), yielding the lowest MAE and RMSE values and the highest R^2 scores across all models. CGCNN provides the best overall test metrics (MAE = 0.230, RMSE = 0.406, and R^2 = 0.775) with good generalization, while XGBoost achieves nearly identical consistency between training and testing sets. GCN, MPNN, and GP display less stability, with modest train-test discrepancies and lower overall performance. The models performed slightly better for cationic surfactants than with nonionic surfactants, despite having roughly half as many datapoints (n = 224 vs n = 415). The cationic surfactants exhibit lower MAE and RMSE values but have consistently lower R^2s, indicating a weaker ability to capture underlying structure-property relationships. The CGCNN model showed noticeable overfitting for this category (MAE of 0.252 for training and 0.333 for testing), whereas GCN demonstrated the most consistent performance, and XGBoost and GP exhibited moderate declines from training to testing, with MPNN showing the best generalization with slightly better performance for the testing set. Nonionic surfactants produce the most balanced performance across all models. CGCNN again has the best performance (MAE = 0.260, RMSE =0.381, and R^2=0.908), followed closely by MPNN (MAE = 0.302, RMSE = 0.421, and R^2 = 0.887), both showing strong

generalization. GCN, XGBoost, and GP all performed poorly for the nonionic surfactants compared to the other two models.

The performance of the models across the three headgroups is not solely a result of data imbalance (cationic 20.2%, anionic 42.4%, nonionic 37.4%). Intrinsic chemical differences within each category of the dataset also contribute. Cationic surfactants (e.g., quaternary ammonium with Cl^-/Br^- counterions in the dataset) prove most challenging for ML, due to counterion interactions, flexible linkers, and conformational variability, which descriptors and graph-encodings cannot describe, yielding lower R^2 despite comparable MAE/RMSE to nonionics, causing deficits vs. anionics/nonionics. Anionic surfactants make up the largest share of the dataset and are relatively uniform, with headgroups dominated by sulfates, sulfonates, phosphates, and carboxylates; this consistency makes their CMC values easier to correlate with polarity- and surface-related descriptors. Nonionic surfactants are more structurally diverse and are therefore moderately more difficult to predict.

The work in this chapter demonstrates that the three GNNs (CGCNN, GCN, MPNN), particularly CGCNN, provide the most accurate predictions of CMC across cationic, anionic, and nonionic surfactant headgroups, achieving MAEs of 0.26-0.34 $log(M)$ on test sets. These GNNs effectively incorporate temperature as a graph feature and generalize well, with MPNN showing no overfitting while CGCNN and GCN exhibit only slight overfitting mitigated by the larger dataset size, CV, and regularization. They outperform descriptor-based XGBoost (MAE 0.360 $log(M)$) and GP (MAE $log(M)$)

models in accuracy despite XGBoost excluding temperature from its top 71 features due to low correlation and offering 8,500× faster training enabling rapid high-throughput screening of surfactant candidates as a faster complement to time-intensive experiments or less generalizable simulations like COSMO-RS that takes 10-60+ times longer than pure ML, not including setup.[321] The near-normal error distributions and minimal temperature-error correlations across all models where error patterns mirror data density rather than inducing biases confirm their robustness for practical surfactant design applications.

Conclusion

The work presented in this chapter demonstrated that ML models can accurately predict the CMC of cationic, anionic, and nonionic surfactants across different temperatures and in a more high-throughput manner compared to traditional experimental and computational methods. The GNNs outperformed both XGBoost and GP; however, this performance improvement came at the cost of longer runtimes, from seconds to several minutes. In general, the GNNs performed the best, with CGCNN exhibiting the best metrics of all ML models studied. CGCNN did have some minor overfitting, even with strategies to limit the overfitting, such as L2 regularization and early stopping. Although these models are not as precise as experimental techniques, they offer a significant advantage: they are faster. The ML models were able to achieve a mean error of 0.3-0.4 $log10(M)$, with the best model (CGCNN) having an MAE of 0.261 $log10(M)$ and also being the most precise with an RMSE of 0.428 $log10(M)$. The accuracy, precision, and speed of the three GNNs make them practical options for a high-throughput screening tool

for CMC across different headgroups and at varying temperatures. The work in this chapter studied the impact of temperature on the model predictions. Encoding temperature into the graph data, a previously unexplored approach that did not negatively affect performance, with the same being said to incorporating temperature into the feature vector for RDKit-derived molecular descriptors for the XGBoost and the GP models. All five models successfully leveraged temperature as a predictive factor, improving accuracy without overfitting the most frequently occurring temperature range in the dataset (20-30°C). These models thus provide a practical, innovative approach for fast CMC prediction that works across different surfactant headgroups and temperatures. The chemical significance lies in enabling high-throughput screening of surfactant libraries for RFB microemulsions, where CMC governs micelle-induced changes in ion mobility and electrode wetting that directly impact power density unattainable through labor-intensive techniques like surface tensiometry or conductivity titration. Encoding temperature directly into graph node features or descriptor vectors preserved performance across 20-80°C without overfitting the common 20-30°C range, demonstrating robust generalization for practical formulation under operating conditions.

CHAPTER 4:

COULOMB MATRIX APPROACH FOR PREDICTING MELTING POINT OF ORGANIC MOLECULES

Abstract

Accurately predicting MPs is a difficult problem in materials science and chemistry. The MP influences applications ranging from drug formulation to energy materials design. Despite the importance of predicting the MP, it remains difficult to model due to the complex interplay between molecular structure, intermolecular forces, and crystal packing effects. Reliable prediction methods could accelerate the discovery of materials by reducing experimental costs and narrowing the search space for suitable compounds. Previous efforts to use ML have explored a range of models and molecular representations with limited success. In the work in this chapter, CMs are used as model inputs to enhance the predictive performance of ML models. Using three ML models, XGBoost, LightGBM, and CatBoost, the proposed ML pipeline delivers fast and accurate MP predictions for organic molecules, exhibiting a 29.9% MAE improvement over existing ML frameworks.

I thank one of my students, Daniel Joy, for substantial assistance with the implementation, optimization, and testing of the codebase used for the work in this chapter. The conceptual design, research questions, and interpretation of results remain my own.

Introduction

The melting point MP is a fundamental physical property that has been studied as far back as 4,000 BC, when copper was smelted to create early tools.[322] From early metallurgy to modern science, understanding melting behavior has remained central to both practical applications and scientific discovery. A substance melts when thermal energy overcomes intermolecular forces holding its solid lattice. Hydrogen bonding and polarity create directional networks, elevating MPs, while molecular rigidity enhances packing efficiency, both of which dominate over van der Waals forces in organics. The MP plays a critical role across materials science,[99,323–325] energy applications,[326,327] and pharmaceuticals.[328–331]

Predicting the MPs of complex molecules remains challenging for ML for several reasons. First, commonly used descriptors, such as those generated by RDKit, primarily encode local molecular structure. In contrast, MP relies heavily on crystal packing and electrostatic interactions, which are not fully captured by such descriptors.[103,215] Second, many datasets lack reliable measurements at extreme temperatures, limiting model generalization. Third, existing models are often trained on narrow subsets of compounds, such as molecules containing only C, O, and N atoms, and inconsistent handling of melting point ranges can introduce noise into target values. [99,103,332]

Recent advances have explored curated datasets and improved molecular representations. For example, Zhu et al. used the CCDC dataset (~59K structures) with

RDKit 2D descriptors. They achieved an MAE of 28.1°C within drug-like ranges (50-250°C), finding 2D descriptors more robust than 3D crystal features due to conformational uncertainty.[252] Alternative approaches include NLP-based SMILES representations (Mi et al., MAE 27.88°C)[100] and graph convolutional models (Coley et al., MAE 26.18°C),[165] which learn task-specific structural features. Larger-scale studies, such as Tetko et al. (228,173 compounds), emphasize the importance of dataset size, achieving an RMSE of 32°C using NNs.[333] Finally, Galeazzo and Shiraiwa improve isomer resolution and functional group sensitivity, reaching an MAE of 26.2°C.[99] Despite these advances, most studies rely on relatively small or chemically narrow datasets, limiting generalization across diverse molecular spaces.

Despite these advances, MP prediction remains unresolved due to three key limitations. First, existing descriptors fail to encode three-dimensional packing geometry adequately. Traditional representations, such as RDKit descriptors and ECFPs, capture 2D topology and composition but do not explicitly model interatomic distances R_{ij} or nuclear charge interactions Z_i (approximated using atomic number), which influences crystal lattice formation. Second, many datasets lack chemical and thermal diversity, focusing primarily on H/C/O/N compounds within narrow temperature ranges (50-250°C), while excluding broader chemistries and extreme MPs. Third, current models underutilize physically meaningful representations of electrostatic interactions, limiting their ability to capture structure-property relationships. The work in this chapter addresses these gaps

using CM representations, which encode nuclear repulsion terms $\frac{Z_i Z_j}{R_{ij}}$, enabling gradient boosting models to infer intermolecular packing effects better.

To address these challenges, a large dataset of 539,022 organic molecules with SMILES strings and experimentally measured MPs spanning 0-300°C is compiled. A comprehensive feature set including 59 RDKit and PaDEL descriptors (e.g., TPSA, number of rotatable bonds), 2,048-bit ECFPs, and, most notably, 14 CM-based features was compiled. These include the top ten Coulomb matrix eigenvalues (CMEs), as well as statistical summaries (mean, standard deviation, minimum, and maximum values) of the full $N \times N$ matrices. Three GTB models, LightGBM, XGBoost, and CatBoost, were optimized using grid-search CV and then trained for high-throughput MP prediction.

The MP dataset was compiled from multiple sources: 7,522 molecules from the CRC handbook,[334] molecules from the Jean-Claude Bradley dataset (28,645 molecules)[332] and the Bergstrom dataset (277 molecules),[331] and additional MP data obtained from PubChem (12,960 molecules),[197] and OChem (402,000 molecules).[206] A separate dataset from OChem contained 275,133 molecules previously compiled by Tetko,[333] combining all sources and filtering duplicates between them yielded 554,156 unique molecules containing only SMILES strings, MP, and molecule names. Due to the different ways data was stored across sources, the combined dataset only contains the SMILES string, MP, and molecule name for each molecule. Only molecules that have an MP between 0ºC and 300ºC were included in the final dataset, reducing the number of molecules to 539,022. The

removal of data outside of this range reduced the dataset size by 2.73%, ensuring the outlier MP does not negatively affect the results. For molecules with reported MP ranges, the midpoint of the range was used to maximize data retention while capturing the most reliable central estimate. Using the midpoint maximizes the number of values in the dataset while ensuring that each value contains the most reliable value within its given MP range.

To represent the MP data, CMs are represented as $N \times N$ matrices where N is the number of atoms introduced by Rupp et al. for ML for molecular properties.[233] These fixed-size representations encode pairwise intranuclear electrostatic interactions by populating entries with 3D interatomic distances and nuclear charges, projecting spatial nuclear repulsion patterns into a form suitable for tabular models such as GBTs. Alternatives such as Behler-Parrinello symmetry functions, SOAP descriptors, or message-passing graph neural networks (MPGNNs) excel for dynamics or equivariant tasks. However, CMs offer computational efficiency for such a large dataset, better leveraging the speed advantage of ML, without requiring optimized geometries or graph preprocessing. The calculations to generate CMs are defined in equation 1.3,

Figure 4.1 shows three CMs that represent the wide chemical space of the dataset: A, phenol, a methane-like hydrocarbon; B, catechol, an ethanol-like polar organic; and C, Ferrocene, a sparse ring system.

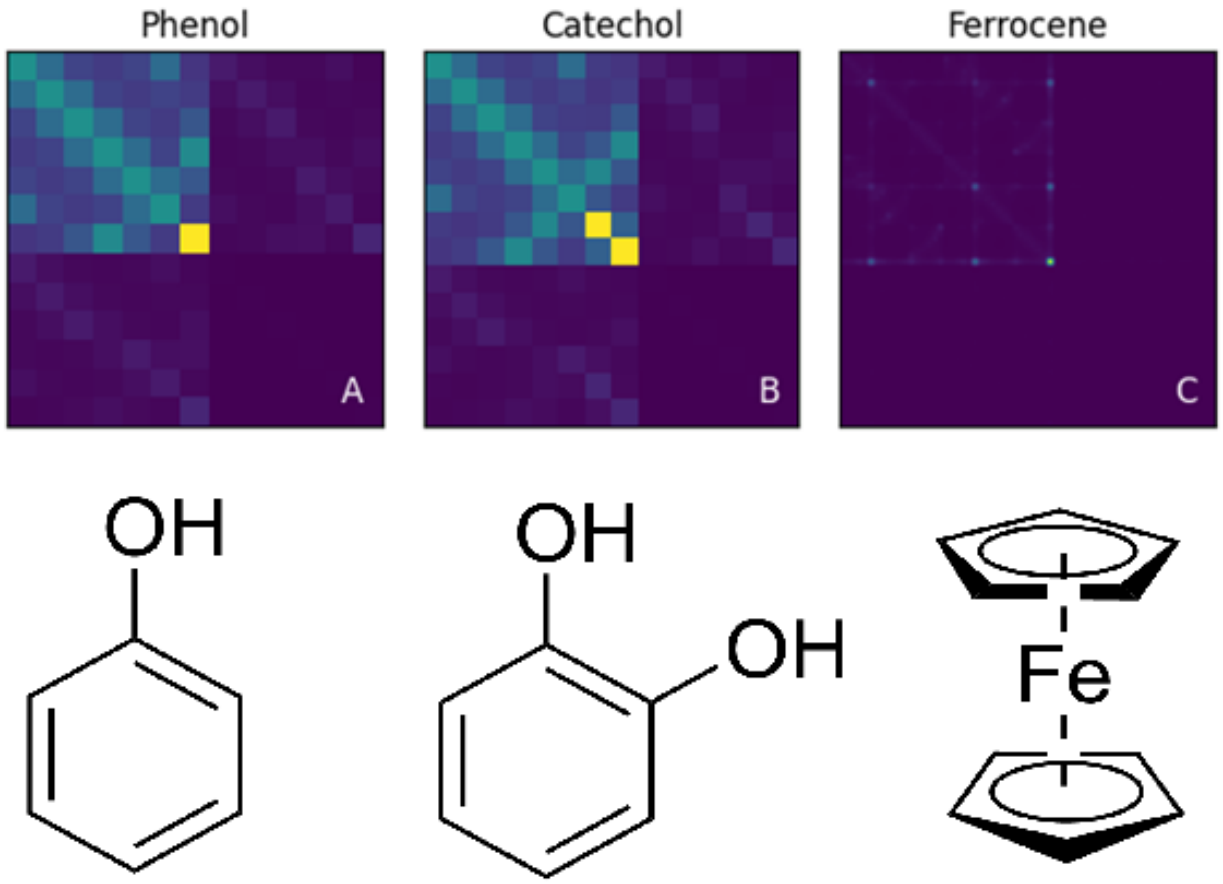

Figure 4.1. Unprocessed Coulomb matrix examples with their structures for A) Phenol, B) Catechol, and C) Ferrocene.

After resizing and flattening the CMs, the CM statistics plus the CMEs reduced prediction error. Resizing variable-size matrices distorts pairwise electrostatic interactions, redundantly capturing the same global patterns already in these 4 statistics. Using just the lower diagonal, CMEs, and CM stats proved more effective than using just the original CM. The eigenvalues of the CMs were computed, selecting the top ten largest eigenvalues as a fixed dimensioned vector. Ten was chosen because it matches the smallest atom count of the molecules in the dataset, ensuring fixed dimensionality while encompassing all spectral information across every molecule. The CM-based descriptors have permutation invariance, meaning they have identical values regardless of SMILES atom ordering while compressing full matrices into dominant electrostatic modes. Rupp et al. and Schrier et al. established this approach for robust molecular featurization.[233,335]

Beyond the CM-based representations, two complementary feature types that capture different aspects of molecular structure critical for MP prediction were added. First, 59 physical and structural descriptors from RDKit and PaDEL (listed **in Table 4.1**), key examples like topological polar surface area (TPSA) quantifying O/N surface area that controls polarity and crystal polarity matching; H-bond donors/acceptors, counting groups forming strong directional lattice networks that elevate melting points; and molecular weight scaling van der Waals contact area and lattice energy. Second, a 2,048-bit ECFP which encodes substructural fragments (e.g., aromatic rings, alkyl chains) as binary bits, capturing repeating motifs that dictate solid-state behavior across chemical families. The combination of traditional and CM-based features yields 2,121 total descriptors across the

Table 4.1. Selected RDKit and PaDEL descriptors for MP predictions.

RDKit	PaDEL
MolWT, TPSA, LogP, HBD, HBA, RotBonds, and MaxPartCharge	AMW, XLogP, CrippenLogP, MLogP, RotBFrac, HybRatio, nRing, nAromBond, nHeavyAtom, nAtom, SpMax_D, SpDiam_D, EE_D, PetitjeanNumber, GATS1e, GATS2e, GATS3e, GATS4e, GATS1p, GATS2p, GATS3p, GATS4p, MATS1e, MATS2e, MATS3e, MATS4e, MATS1p, MATS2p, MATS3p, MATS4p, ATS1e, ATS2e, ATS3e, ATS4e, ATS1p, ATS2p, ATS3p, ATS4p, AATS1e, AATS2e, AATS3e, AATS4e, AATS1p, AATS2p, AATS3p, AATS4p, SpAD_D, SpMAD_D, SpMax_Dt, Kier1, Kier2, Kier3

539,022 molecules in the dataset. The dataset was then normalized using a quantile transformer with a uniform output shape fitted onto the data. No transformation was applied to the target values. The final dataset contains 2,212 descriptors for 539,022 organic molecules with known MP values. The entire data collection and generation process is shown in **Figure 4.2**.

To identify which features drive melting point predictions, SHAP (SHapley Additive exPlanations) analysis was performed using a GTB surrogate model. SHAP values measure the marginal contribution of each feature to individual predictions versus the dataset average. Positive SHAP values (red) increase prediction values, whereas negative SHAPE values (blue) decrease prediction values. **Figure 4.3** shows the mean absolute SHAP values (x-axis impact magnitude) for top features, ranked by total influence, and colors indicating feature value ranges.

The ECFP fingerprints dominate SHAP contribution (55.4% total model output variance), capturing structural motifs consistently across all molecules. H-bond donors rank first among the descriptors (16.0% of model output variance), counting groups such as -OH, -NH that form strong lattice networks, elevating MP values. Lastly, for the descriptors, TPSA ranks second with 15.4% of model output variance, measuring O/N surface area that controls crystal polarity matching and intermolecular attraction strength. The CMs, CMEs, and CM statistics rank fourth with 4.9% of model output variance. TPSA works by quantifying polar surface: high TPSA molecules form H-bonded networks resisting melting; low TPSA (blue) pack via weaker van der Waals. H-bond donors directly

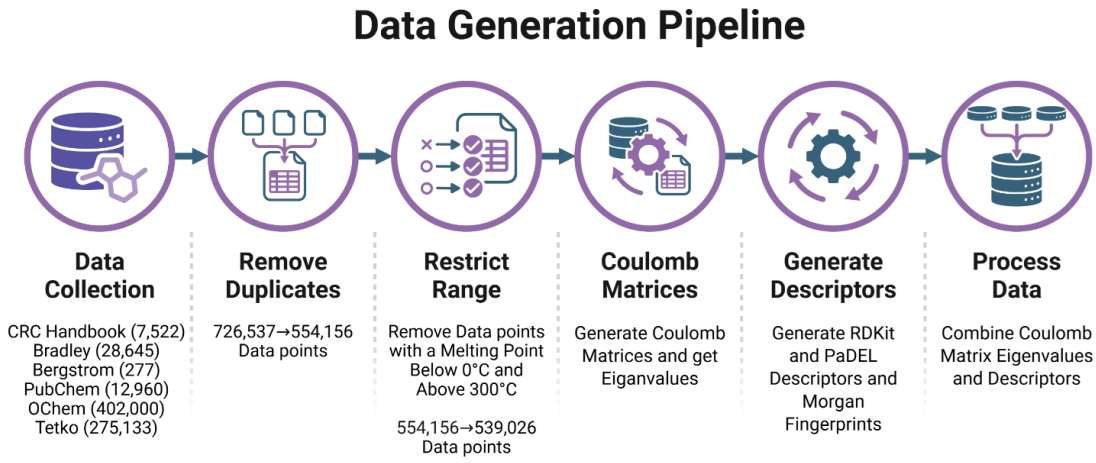

Figure 4.2. Data collection and generation workflow, including duplicate removal, Coulomb matrix generation and processing, RDKit and PaDEL descriptor generation, and Morgan fingerprint generation.

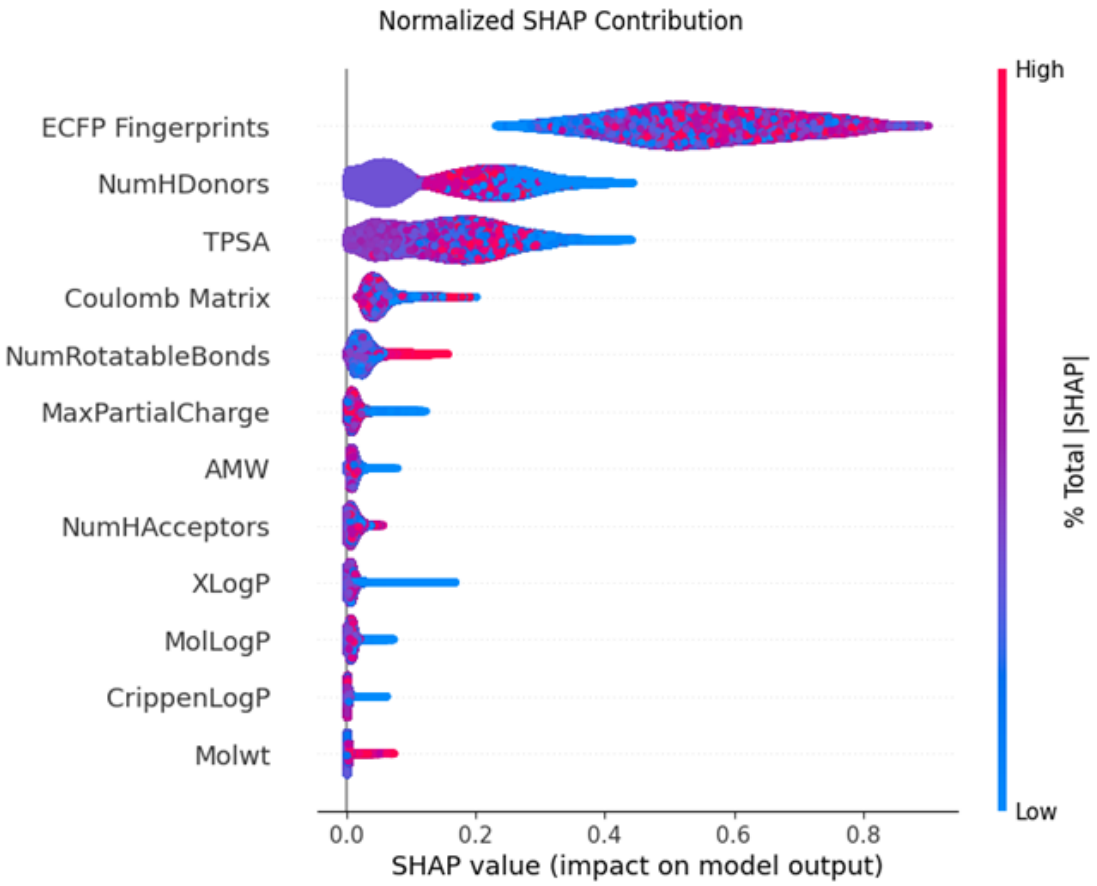

Figure 4.3. SHAP values (red-blue gradient) of the top 12 features, with CMs and ECFPs aggregated.

count lattice-forming groups. ECFPs encode substructures (aromatic rings, alkyl chains) that dictate π-stacking or chain entanglement. CMEs, despite modest average rank, excel on prediction outliers encoding 3D electrostatic geometries where packing anomalies occur.

To more directly measure the impact of the CM-derived representations, the same pipeline using five versions of the dataset was utilized; first with only the ECFPs and descriptors, second with only the ECFPs, descriptors, and raw CM values, then only the raw CM values, CMEs, and CM stats, and finally, the full dataset with ECFPs, descriptors, raw CM values, CMEs, and CM stats.

Methods

To compare how the data representation can be used to predict MPs, three GBT models: eXtreme Gradient Boosting (XGBoost), LightGBM, and CatBoost were compared. GBTs build predictions through sequential decision trees where each new tree corrects errors from the previous ensemble, naturally capturing nonlinear feature interactions critical for MP prediction.[257,336] Tree-based GBTs outperform NNs with structured tabular data such as the dataset because they handle heterogeneous scales, missing values, and complex interactions without extensive preprocessing. Whereas NNs struggle with differently scaled data and require imputation strategies that GBTs handle natively.

The most common GTB variant is XGBoost, which implements regularized gradient boosting with L1/L2 penalties to prevent overfitting. Starting from a base prediction, XGBoost evaluates all possible feature splits across trees in parallel, selecting optimal splits via second-order gradient approximations. The regularization and parallelization abilities make XGBoost robust for high-dimensional datasets while maintaining prediction accuracy.[180]

LightGBM accelerates training through two innovations: Gradient-based One-Side Sampling (GOSS) retains data with large gradients (informative samples) while randomly sampling small-gradient data, and exclusive feature bundling that groups mutually exclusive features to reduce dimensionality. These optimizations make LightGBM faster than XGBoost without sacrificing performance.[246]

CatBoost excels with categorical and mixed data through ordered boosting and has native categorical handling without one-hot encoding, ideal for the ECFP descriptors. Symmetric tree construction ensures stable splits, while automatic feature combinations capture higher-order interactions, enhancing generalization across diverse molecular structures.[247]

Grid search CV optimizes model hyperparameters by testing all combinations from a predefined parameter grid, with CV providing performance estimates for each combination to avoid overfitting during selection. For each grid point, K-fold CV splits the training data into K equal folds, training K surrogate models where each fold serves once as testing while the other K-1 folds are used for training. The mean validation metric

(MAE) across the K folds determines the score of each combination. The best-performing hyperparameter combination is used for each model.

After selecting the optimal hyperparameters, K-fold CV is used to provide unbiased estimates of model generalization by partitioning the full dataset into K equal folds in the same manner as grid search CV, but to train the models rather than generate optimal hyperparameters. The data is randomly split into five folds. The process iterates five times, training the model with four folds while the remaining fold is used for testing, yielding five independent test predictions with a single test result for each data point. The K-fold CV process increases the reliability of the results and allows the entire dataset to be treated as the testing set, referred to as the CV test set. While both grid search and K-fold use CV, each is used for a different purpose; grid search CV finds hyperparameters only, whereas the K-fold CV is used to train and evaluate the models using the optimal hyperparameters. Final performance metrics (MAE, RMSE, R^2 in **Table 4.1**) average these CV test results with their range across the five folds. Finally, to evaluate model performance on a novel dataset distinct from the MP dataset, 18 azobenzene anolyte derivatives from Banik et al. were evaluated.[17] These molecules are not included in the MP data compiled and provide an unbiased evaluation of the models.

Results and Discussion

The testing performance for the three models is shown in **Table 4.2**, which was measured using MAE, RMSE, and R^2 score, each with their variance across folds using each dataset state. Each result represents the aggregate model, combined from the five

Table 4.2. CV testing results for each of the three models; LightGBM, XGBoost, and CatBoost, evaluated using MAE, RMSE, and R^2.

State	Model	MAE	RMSE	R^2
1	LGB	25.72 ±0.05	33.78 ±0.10	0.681 ±0.002
1	XGBoost	24.42 ±0.05	32.26 ±0.10	0.709 ±0.002
1	CatBoost	26.35 ±0.05	34.45 ±0.09	0.668 ±0.002
2	LGB	25.82 ±0.07	33.90 ±0.08	0.678 ±0.002
2	XGBoost	24.55 ±0.04	32.41 ±0.09	0.706 ±0.002
2	CatBoost	26.40 ±0.04	34.51 ±0.07	0.667 ±0.002
3	LGB	21.14 ±0.07	28.21 ±0.12	0.777 ±0.002
3	XGBoost	18.43 ±0.04	25.36 ±0.07	0.820 ±0.001
3	CatBoost	26.05 ±0.05	34.09 ±0.08	0.675 ±0.002
4	LGB	21.16 ±0.05	28.22 ±0.07	0.777 ±0.001
4	XGBoost	18.44 ±0.05	25.35 ±0.08	0.820 ±0.001
4	CatBoost	26.04 ±0.06	34.08 ±0.09	0.675 ±0.002
5	LGB	32.57 ±0.07	41.53 ±0.07	0.517 ±0.001
5	XGBoost	32.10 ±0.08	41.04 ±0.09	0.528 ±0.001
5	CatBoost	33.94 ±0.09	42.88 ±0.07	0.485 ±0.001

individual folds obtained in CV, for that framework. **Table 4.2** demonstrates that all three GTB models showed consistent generalization with negligible variation across folds and all five dataset states. Across every state, all GTBs exhibit consistent results, with minimal variance between folds, indicating stable generalization. XGBoost consistently outperforms the other models, achieving lower MAE and RMSE and higher R^2 values for each state. LightGBM follows closely, while CatBoost, despite its computational efficiency, yields the weakest predictive performance overall.

Performance varies notably across dataset states, reflecting differences in feature representation. States one and two show moderate performance, with R^2 values around 0.67 to 0.71, indicating limited expressiveness of features. In contrast, States three and four demonstrate a substantial improvement in all metrics, with XGBoost reaching an R^2 of 0.82, alongside significantly reduced MAE and RMSE. Finally, State five exhibits a clear degradation in performance across all models, suggesting that the feature configuration in this state is less suitable for capturing the underlying structure of the data. Despite their similar performance, State four is the preferable choice due to its inclusion of CM-based representations. While these features do not significantly shift aggregate metrics compared to state three, they introduce a more physically grounded encoding of molecular structure, capturing interatomic interactions and spatial relationships that are not fully represented by descriptor-based or ECPF features alone. The traditional descriptors added physical interpretability, improving the robustness of the model and enhancing its potential for generalization to chemically diverse systems. In this context, selecting State four provides

a better balance between predictive accuracy and meaningful feature representation, making it the more reliable choice for downstream applications. All subsequent results are evaluated exclusively using the state four features.

Figure 4.4 depicts predicted versus actual MPs for LightGBM, XGBoost, and CatBoost using K-fold CV test folds (black dots), model fit lines (red), ideal fit lines (blue), and 95% prediction intervals (dashed gray). The 95% confidence intervals (shaded blue) are narrow and, as a result, barely visible, indicating high precision across models. The trends match **Table 4.2**, with XGBoost performing best as its fit line is closest to the ideal fit, and its prediction interval is tightest, clustering predictions near the fit line. All models overpredict lower values at below ~150°C and underpredict above ~150°C; this is a common issue in MP regression literature that large datasets such as ours mitigate but do not fully resolve; XGBoost shows the mildest bias, followed by LightGBM, then CatBoost.

The trends in **Figure 4.4** reveal an intriguing reversal for the 18 azobenzene anolyte derivatives (red dots) from Banik et al. CatBoost performs best overall, with its predictions being most in line with the true values, while XGBoost shows significant underprediction for every datapoint. LightGBM performs similarly to CatBoost, although slightly worse, but still outperforms XGBoost. The results for the azobenzene anolyte derivatives oppose the trends on the compiled MP dataset, where XGBoost has the best results, highlighting how aggregate performance masks subset-specific behaviors. The discrepancy between the compiled MP dataset and the azobenzene anolyte derivatives can be attributed to the azobenzene anolyte derivatives containing wavy bonds, which SMILES cannot represent,

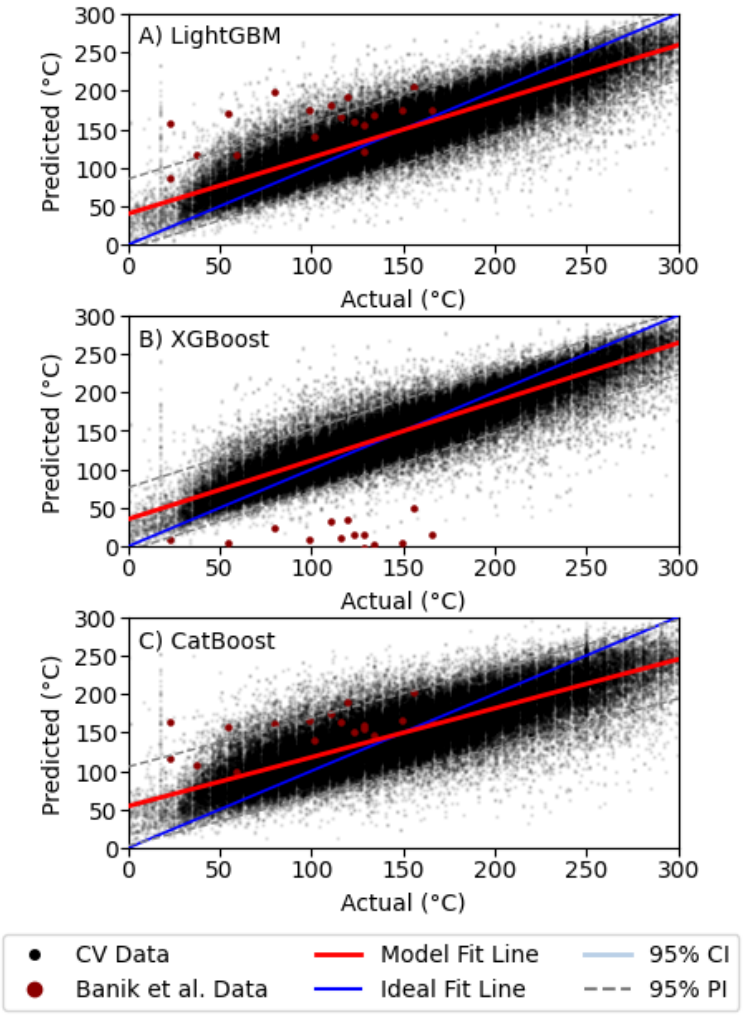

Figure 4.4. Regression plots for MP predictions for A) LightGBM, B) XGBoost, and C) CatBoost. CV test points are shown as black dots, Benik et al. evaluation points as red dots, the fit line of each model as a red line, the ideal fit line as a blue line, the 95% prediction interval as gray dashed lines, and the 95% confidence interval as shaded blue.

and, are treated as carbon bonds. Additionally, because each of the azobenzene anolyte derivatives is similar to the others, the models cannot distinguish between them well.

Figure 4.5 displays kernel density estimates (KDEs) of prediction errors for the three GTBs over their CV test folds, with medians as dashed gray lines near 0°C offering a distribution view resilient to outliers. XGBoost exhibits the tightest distribution with a standard deviation (std) of 25.25°C, demonstrating the highest density near-zero, and a mean of -0.08°C, confirming it as the best model. LightGBM with a std of 28.12°C and a mean of -0.05°C, finally, CatBoost has the largest std of 33.98°C, and a mean of -0.09°C. All show slight negative skewness (-0.141 for XGBoost, -0.116 for LightGBM, and -0.135 for CatBoost), indicating minor underprediction tails but Gaussian-like cores peaked at near zero. The superior symmetry and precision of XGBoost align with **Table 4.2** and **Figure 4.4**, as narrower std drives lower losses despite shared medians near zero. Unlike MAE, which is outlier-sensitive, these KDEs reveal expected error probability. XGBoost delivers the most reliable predictions with zero error, shown by its tallest peak and tightest spread among the three distributions.

Figure 4.6 illustrates the training (solid) and validation (dashed) loss over iterations for each model, highlighting convergence speed, stability, and overfitting behavior during K-fold CV, averaged across all folds. LightGBM and XGBoost both show steadily decreasing training and validation MSE across the full iteration range, with XGBoost ultimately reaching the lowest validation loss, while the RMSE of the predictions made by

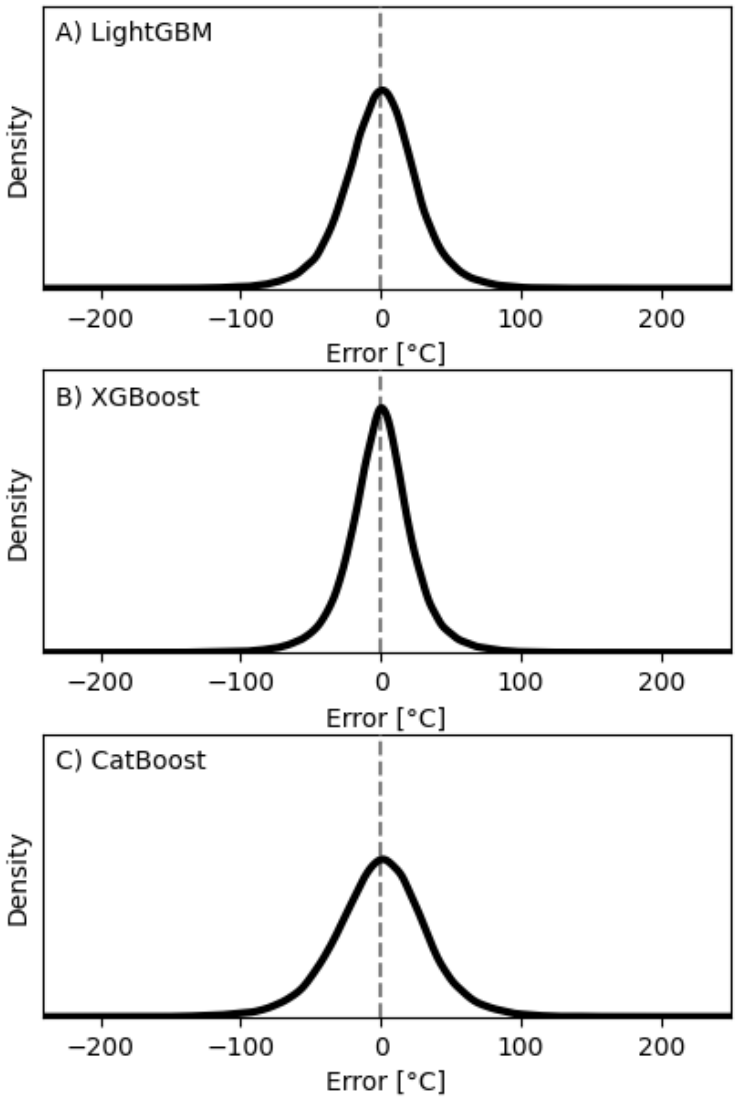

Figure 4.5. Error distributions for A) LightGBM, B) XGBoost, and C) CatBoost shown as density as a function of error in ºC.

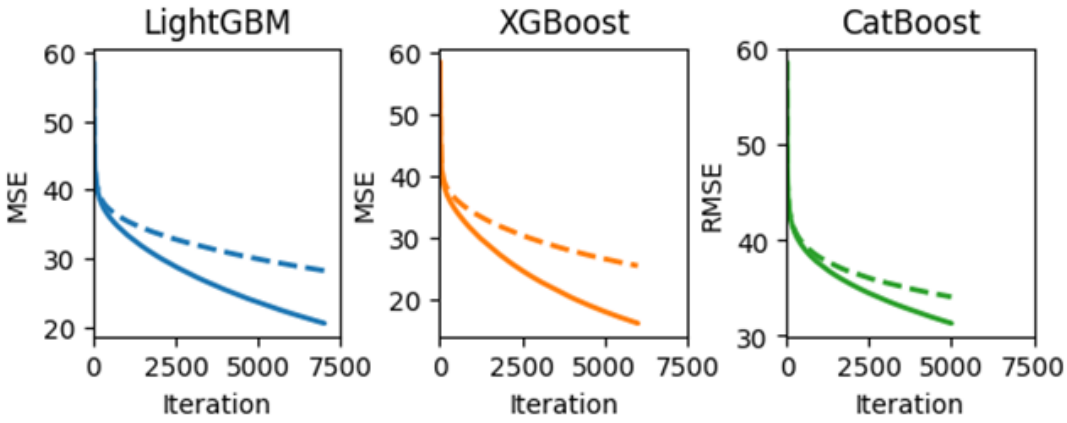

Figure 4.6. Training and Validation loss curves for A) LightGBM, B) XGBoost, and C) CatBoost.

CatBoost curves level off much earlier and remain above 30°C. The smaller gap between the training and validation curves for CatBoost indicates that it overfits least, but the early plateau explains why it performs worse overall than LightGBM and XGBoost, which trade a slightly larger gap for substantially lower final error. However, the smaller gap could also be a result of using a different loss function. Grid search CV found that MSE is the best objective function for LightGBM and XGBoost, and RMSE is best for CatBoost. For all three models, early stopping did not trigger, meaning they trained for the full iteration range. The lower convergence of XGBoost without excessive overfitting makes it the optimal choice for this MP prediction task.

Finally, **Figure 4.6** illustrates training (blue) and validation (orange) loss over iterations for each model, demonstrating convergence speed, stability, and overfitting level during 5-fold CV, averaged over all folds. During grid search CV, MSE was found to be the best objective function for LightGBM and XGBoost, and RMSE was best for CatBoost. All three models begin to converge in under 1,000 iterations and continue to improve until early stopping is triggered. LightGBM and XGBoost exhibit near-identical trends, although LightGBM needed fewer iterations to converge. CatBoost, which used RMSE unlike LightGBM and XGBoost, did not trigger early stopping and had the smallest degree of overfitting, despite the worse results.

Overall, XGBoost emerged as the superior model for MP prediction, achieving an MAE of 18.42°C, RMSE of 25.32°C, and R^2 of 0.821 across 5-fold CV using the augmented CM-

based dataset a 29.7% MAE improvement over state-of-the-art methods commonly in the 26°C-33°C range. These results demonstrate that carefully engineered CM representations paired with optimized gradient boosting can achieve accurate high-throughput MP predictions.

A comparison of the results presented with other MP prediction studies is shown in **Table 4.3**, which demonstrates that the CM-based approach provides the lowest MAE and RMSE in MP predictions for ML models, achieving 18.42°C MAE and 25.32°C RMSE on the largest general dataset (539,022 molecules), surpassing graph embeddings (Coley: 26.18°C MAE), NLP (Mi: 27.88°C), and large-scale NN efforts (Tetko: 32°C RMSE). Additionally, it shows that dataset size alone does not determine predictive performance; smaller curated sets yield higher R^2 via low variance, but the broad chemistry of the large dataset generalizes better. While comparisons are not 100% comparable due to differences in descriptors, MP ranges, and evaluation splits, benchmarking against other general datasets advances this heavily studied problem. Accurate MP prediction accelerates materials screening in fields where MP often detects bottlenecks discovery, allowing rapid selection without experiments, enabled by the 29.9% reduction in MAE. Notably, even the second and third best feature states outperform prior work, confirming the broad value of CM-augmented descriptors.

By compiling the largest MP dataset to date with 539,022 organic molecules covering 0-300°C MP range to tackle the chemical gaps in earlier studies, limited to narrow H/C/O/N compounds. Using CM descriptors that better capture the electrostatic packing

Table 4.3. Comparison of ML model results using MAE, RMSE, and R^2 using the testing set or equivalent to predict MP with the best results bolded.

Method	Dataset Size	MAE	RMSE	R^2	Paper
Our Work	**539,022**	**18.42**	**25.32**	0.821	Our Work
MOE Descriptors	287	33.8	48.1	0.50	Hughes et al.[103]
ECFP Embeddings	~220,000	31.0	40.1	0.77	Galeazzo & Shiriwa[99]
RDKit	58,810	28.1	-	-	Zhu et al.[101]
NLP	~30,000	27.88	37.35	0.83	Mi et al.[100]
Mult. Descriptors	289,000	-	32	-	Tetko et al[333]
Graph Embeddings	3,019	26.18	35.54	**0.85**	Coley et al[165]

forces that RDKit, PaDEL, and ECFP miss, overcoming key longstanding barriers in MP modeling. Grid search CV tuned hyperparameters for three GTBs, then using K-fold CV to train the final GTBs yielded an MAE of 18.42°C, an RMSE of 25.32°C, and an R^2 of 0.821, 29.9% MAE better than state-of-the-art MP prediction methods despite challenges adjusting the azobenzene anolyte derivatives from Banik et al.

The SHAP analysis reveals that the ECFP substructures represent 55% of the output variance of the surrogate model, followed by HBDs and TPSA with 16% and 15.4%, respectively, and then CM-based features with 5%. The framework presented in this chapter drives CM-based descriptor innovation, delivering the first reliable sub-20°C MAE across a broad range of molecules, turning MP prediction from costly experiments into a fast screening tool for batteries, pharmaceuticals, and materials.

Conclusion

The study demonstrated in this chapter introduced two major advances: first, the largest MP dataset was compiled for ML (539,022 organic molecules spanning 0-300°C), 1.96× larger than Tetko et al.'s prior record. Compiling MPs from the CRC, Bradley, Bergström, PubChem, and OChem datasets. Second, CM-based descriptors complement RDKit/PaDEL/ECFPs, explicitly modeling nuclear repulsions key to packing. XGBoost emerges superior with an MAE of 18.42°C, RMSE of 25.32°C, and an R^2 of 0.821 via grid search CV hyperparameter selection followed by K-fold CV training. This approach outperformed all state-of-the-art benchmarks, including Coley et al. (26.18°C), Mi et al.

(27.88°C), and Tetko et al. (32°C RMSE) by 29.9% MAE. XGBoost is evidently the best model, having the tightest error distribution with a std of 25.25°C with peak density at zero, with predictions best aligning with the true values, and the lowest validation loss convergence without excessive overfitting, outperforming LightGBM and CatBoost. These contributions resolve the core challenges of predicting MPs: data sparsity in chemically narrow datasets, traditional descriptors ignoring packing electrostatics, which are key to MP, delivering reliable screenings in energy storage, pharma formulation, and materials, and augmenting slow experiments with high-throughput computation.

The predicted MPs determine whether candidate oils remain liquid under operating conditions, which is critical for type IV μE stability; data sparsity had previously limited models to simple alkanes and esters, while descriptor-only approaches struggled to capture the electrostatic and packing effects that govern lattice energy. The CM-based approach enables high-throughput screening of redox-active oils, replacing traditional methods with ML predictions.

CHAPTER 5:

GRAPH ENCODERS FOR PREDICTING NON-AQUEOUS MOLECULAR SOLUBILITY

Abstract

ML is widely used to predict molecular solubility, yet accurate ML prediction of non-aqueous solubility remains challenging. Despite growing interest, existing models often generalize poorly because they target narrow chemical spaces, single solvents, or rely solely on predefined descriptors. The work in this chapter combines VAE latent graph embeddings with molecular descriptors, ECFPs, and temperature to model the solubility of non-aqueous solutions. Solutes and solvents are encoded as latent vectors using graph-based VAEs trained using the QM9 and ZINC datasets, augmented with RDKit and PaDEL descriptors, and subsequently used with an XGBoost regressor. Bayesian optimization is used to select the VAE latent dimensionality. Using the BigSolDB dataset, the proposed approach reduced the MAE by 51.2% relative to the current state of the art in solubility prediction. Demonstrating that combining learned latent graph embeddings from VAEs with established molecular descriptors yields a generalizable and accurate method for non-aqueous solubility prediction.

Introduction

Solubility is of fundamental importance across many fields, such as energy storage,[5,6,27,32,33,264] materials science,[337–340] and pharmaceuticals.[338,341,342] At the molecular scale, it reflects a balance of solute-solvent interactions with hydrogen bonding between polar oil headgroups and solvent molecules, dispersion forces stabilizing hydrophobic tails in the oil phase, and electrostatic screening modulated by solvent dielectric constant against

thermodynamic driving forces favoring solute-solvent mixing versus solute-solute or solvent-solvent cohesion. However, measuring solid-liquid solubility across different solvents and temperatures remains experimentally intensive, requiring long times to reach equilibrium and to perform analytical quantification for each solvent-solute pair. The experimental burden motivates accurate solubility prediction. Recent increases in available data resources and computational power have enabled ML methods for this purpose. Predicting solubility is challenging due to (1) data scarcity/noise, (2) molecular descriptor inadequacy, and (3) model selection and architecture, as described by Hughes et al.[103] Early computational approaches relied primarily on physics-based methods such as COSMO-RS and QSPR.[311,337,343–345] The general solubility equation (GSE) as described by Jain et al., can be used to estimate solubility, although it often falls short of computational and ML methods. These methods are often paired with models that map fragment-based descriptors to solubility in specific chemical domains (e.g., drug-like or agrochemical compounds).[343,346] More recent hybrid workflows integrate COSMO-RS-derived quantities as inputs to ML models to improve performance across broader sets of organic solvents.[347]

ML has been used to predict solubility in both non-aqueous and aqueous solutions, using tree-based ensemble methods and trained on molecular descriptors or extended-connectivity fingerprints (ECFPs). However, these methods remain constrained by the limitations of conventional descriptor-based representations.[345,348–350] Consequently, ML studies have sought to move beyond fixed molecular descriptor sets by using more

expressive representations and models that capture a wider range of complex structure-property relationships. Gaussian process (GP) approaches have been used to predict intermediate quantities such as Hansen solubility parameters (HSPs) while also providing uncertainty estimates useful for screening.[190,337]. Solubility modeling has also been pursued in classification settings (e.g., soluble vs insoluble), using CNNs and Bayesian classifiers trained on structural or physicochemical molecular descriptors.[348,351]

In parallel with these modeling advances, application-driven contexts such as RFBs face solubility as a key performance bottleneck, motivating high-throughput experimental pipelines that generate datasets amenable to ML-driven screening.[340] Nevertheless, generalization is often limited when ML models are trained on narrow chemical domains, limited solute or solvent coverage, or representation schemes that do not widely capture solute-solvent structure.

The work in this chapter, a scalable ML pipeline for predicting non-aqueous solubility, is developed using the ZINC and QM9 datasets, training three VAEs to generate latent vector-based representations.[198,199] Solutes and solvents are represented as molecular graphs and then encoded by the VAEs into fixed-dimensional latent vectors. The VAE latent vectors are augmented with temperature, conventional molecular descriptors obtained with RDKit and PaDEL, and Morgan fingerprints, a type of ECFP. The resulting descriptor set is then used to train an XGBoost regressor to predict solubility. The BigSolDB dataset (103,944 entries with solute, solvent, and temperature information) is used to train the XGBoost model.[104] The XGBoost model is further evaluated using a

dataset of 21 solutions.[8] The presented hybrid design using CGVAE with XGBoost couples the expressiveness of learned graph embeddings with conventional molecular descriptors and the accuracy and interpretability of gradient-boosted trees, enabling large-scale screening across diverse solute-solvent pairs.

Methods

AEs are NNs trained to learn a compact, lower-dimensional representation of data and to reconstruct the original input from that representation. An encoder maps the input to a latent representation, and a decoder maps that latent representation back to the original input space.[166,239–242] In practice, the encoder performs this compression through successive layers that reduce the representation's dimensionality. The decoder then applies successive layers to reconstruct the input, and training optimizes both encoder and decoder parameters to minimize reconstruction error. The latent vector is the output of the most compressed layer and serves as the interface between the encoder and decoder.

A VAE is an autoencoder whose encoder outputs a probability distribution over latent variables, and whose training objective includes a KL divergence term that regularizes the latent space.[166,221,352,353] The VAE's probabilistic formulation typically yields a smoother latent space than an autoencoder and is therefore well suited to complex inputs such as variable-size molecular graphs. The encoder outputs two parameter vectors, typically a mean and a variance (or log variance), that define the latent distribution from which a latent vector is sampled. The smoothing effect arises from the KL term, which regularizes the latent space so that similar graphs map to nearby regions, even when their

sizes differ.[352] In contrast, a standard autoencoder can learn an irregular, fragmented latent space in which nearby latent vectors do not necessarily correspond to structurally similar molecules, resulting in less reliable representations.

The CGVAE framework implements a molecular graph-based VAE with a standard VAE encoder and latent space, with the decoder replaced using a pure-algorithmic method rather than an NN-based method. The encoder uses a GGNN that maps molecular graphs to latent distributions $(\mu_v,\ \sigma_v)$. Latent vectors z_v are sampled, and trained using reconstruction loss plus KL divergence regularization.[352] The decoder reconstructs molecules by sequentially adding nodes and edges following a breadth-first search-based process, approximated by Monte Carlo sampling. The VAE is trained by minimizing the graph reconstruction loss. The graph reconstruction loss maximizes the log-likelihood (log of a probability function) of generating the observed graph from the initial unconnected nodes. Finally, the graphs are reassembled by iteratively adding the edges to the unconnected graph to form the molecule that can be represented by a valid SMILES.

Finally, the SMILES string of the generated molecule is constructed from the completed graph using the RDKit parser.[215] Unlike a standard VAE, which utilizes solely reconstruction and KL losses as the loss function, the loss function uses reconstruction (sum of node and edge losses using CE) and KL losses with a valency mask, ensuring that only valid molecules are generated; this is not considered part of the loss function.[166]

CGVAE encodes SMILES into molecular graphs, containing atoms as nodes, bonds as edges, and explicit connectivity information, using RDKit.[166,215] The atom feature

vectors contain solely a one-hot encoding of the atomic numbers, and the bond feature vector consists only of discrete bond order, i.e., single, double, triple, or aromatic bond. Allowed valences are defined as external constraints rather than learned features, applied as a mask to enforce chemical validity. Although the CGVAE as a whole will enforce chemical validity upon the generated graphs, the latent space itself has no guarantee to correspond to a chemically valid molecule, which can have a negative effect on the performance of the latent vectors as a molecular descriptor.

BO can efficiently approximate the optimum of a continuous function. In this case, BO seeks an optimal latent vector size without exhaustively evaluating every value within the specified range and using BO to minimize the testing loss as a function of each VAE's latent vector size. BO begins by sampling *n* pseudo-random latent vector sizes and recording their testing losses after training n VAEs, each using one of the *n* sampled sizes. These points are used to build an initial GP surrogate model that estimates the expected loss and uncertainty across the full range. BO then runs m additional iterations; at each iteration, the GP surrogate model is updated, and the next latent vector size is selected using the best LCB. The LCB balances uncertainty (exploration of untested regions) and expected loss (exploitation of sizes that have performed well). After all $n + m$ evaluations, BO selects the latent vector size with the lowest observed testing loss, whether obtained from the pseudo-random or LCB-selected samples, yielding a near-optimal latent size without the cost of exhaustive search over the entire range.

XGBoost is an ensemble tree method related to random forests, but it uses gradient-boosted decision trees. Unlike random forests, which build trees independently and aggregate their predictions, gradient boosting builds trees sequentially, with each tree correcting errors from the previous ensemble. In XGBoost, this framework is augmented by L1 and L2 regularization, native handling of missing values, and CPU parallelization.[354,355] Training iteratively adds trees by selecting splits that minimize a gradient-based objective, applying regularization to control model complexity, and updating predictions via boosting until convergence.

To train the XGBoost, group K-fold CV is used. Unlike standard K-fold CV, group K-fold CV assigns all samples from a given solute-solvent pair to the same fold. The group K-fold CV prevents near-duplicate samples for the same solute-solvent pair at different temperatures from being split between the training and validation sets, reducing data leakage. In each split, the model is trained on K-1 folds, with the remaining fold held out for testing. This process yields more reliable results and allows us to treat each data point as part of the test set. After training, an ensemble of K trained XGBoost models is used to generate predictions on the evaluation dataset from Ahn et al., with the final predictions computed by averaging across the K folds.[8,356]

To train the VAEs to generate latent vectors for the solutes and solvents, two datasets were selected: ZINC and QM9. These datasets were chosen because they contain a large amount of organic molecules, are widely studied, and have been demonstrated to work with the CGVAE framework.[166,198,199] ZINC is a collection of 249,456 drug-like

molecules defined by the Lipinski rule of five: LogP < 5, MolWt < 500 Da, H-bond donors ≤ 5, and H-bond acceptors ≤ 10.[199,357] The molecules in ZINC contain nine types of elements (Br, C, Cl, F, H, I, N, O, S). The second dataset, QM9, contains 133,885 molecules with five elements (H, C, N, O, F). In QM9, the largest molecules contain at most nine heavy (non-hydrogen) atoms, whereas ZINC has no such limit. A combined dataset that used both ZINC and QM9 was used, which is referred to as "Merged," and was used to train a third VAE. For ZINC, the atom encoding takes the form of an atomic symbol followed by the number of bonds, with the charge given in parentheses. The 14 supported types in ZINC are: Br1(0), C4(0), Cl1(0), F1(0), H1(0), I1(0), N2(-1), N3(0), N4(1), O1(-1), O2(0), S2(0), S4(0), S6(0). The encoding structure for QM9 is simpler and uses only the atomic symbol, without explicitly encoding bond count or charge. When merging the datasets, ZINC's encoding structure is used because it is compatible with both ZINC and QM9, whereas QM9's encoding structure is not compatible with ZINC. After encoding, three VAEs are trained, one per dataset. The three VAEs trained on the ZINC, QM9, and merged datasets are referred to as "ZINC-VAE", "QM9-VAE", and "Merged-VAE", respectively, where the name denotes the training dataset and distinguishes the model from the dataset.

To train the XGBoost model to predict solubility, the BigSolDB dataset is used. BigSolDB comprises 103,944 solutions from various journals using the search term "Solubility" with manual filtering, with a total of 1,595 sources. Each data point represents

a unique combination of solute, solvent, and temperature, with a corresponding solubility in molarity (mol/L), obtained experimentally.[104]

However, of the original 103,994 entries, only 83,099 were used after processing with the CGVAE framework discussed in the Models section due to generating invalid graphs.[166] After filtering, the compatibility of the remaining entries with the CGVAE framework is verified. Then, the VAE latent vectors are encoded, and molecular descriptors are generated for solutes and solvents separately, enabling distinct embeddings and descriptors for better optimization. Examining the BigSolDB solutes and solvents (referred to henceforth as BigSolDB-solutes and BigSolDB-solvents), the BigSolDB-solvents are generally smaller and contain atom types compatible with both the QM9 and ZINC encoding schemes. In contrast, BigSolDB-solutes are more complex and include atom types not compatible with the QM9 encoding scheme, so they are compatible only with the ZINC encoding scheme. Accordingly, solutes are evaluated only with the ZINC-VAE and Merged-VAE, whereas the solvents are evaluated with all three VAEs.

To better evaluate the XGBoost model trained on the processed BigSolDB dataset, a set of 21 solutions obtained from table 2 in Ahn et al. is used, henceforth referred to as "Eval", which contains "Eval-solutes" and "Eval-solvents."[8] The Eval dataset is never used during VAE or XGBoost training, ensuring that model development is completed independently of Eval. Consequently, evaluation on this dataset provides an unbiased assessment under real-world conditions. Eval is intended to provide external validation using solutions absent from the BigSolDB-derived training and testing sets. Although Eval

includes chemically diverse solutes, it contains only a few solvents (acetonitrile, dimethylformamide (DMF), and propylene carbonate (PC)); acetonitrile represents 19 of the 21 solutions. In addition, all Eval data points were measured at 25°C, illustrating how evaluation datasets are often small and restricted to narrow chemical domains, reflecting a trade-off between dataset scale and applicability. Nonetheless, Eval provides an external, unbiased, and application-oriented test of the generalization of XGBoost beyond BigSolDB chemical space.

Analyzing the solubility values in BigSolDB shows that they are heavily skewed toward zero, as shown in **Figure 5.1A**, which is undesirable for ML. To address this, a $-log10$ transform is applied to obtain a solubility distribution with a range that more closely resembles a normal distribution, shown in **Figure 5.1B.** The log transformation converts the units used for solubility in BigSolDB from mol/L to $-log(mol/L)$ referred to as $log(S)$, which is more commonly used in ML-based solubility modeling, allowing for more comparable results. Outliers are then removed from BigSolDB using a modified Z-score proposed by Kappal, resulting in the removal of 1.89% of data points.[358] The temperature distribution in **Figure 5.1C** is approximately normal, centered at 25ºC, with a small variance.

Before using the four datasets (ZINC, QM9, BigSolDB, and Eval), removing duplicates ensures that both the VAE and XGBoost model training and subsequent evaluations are not influenced by duplicate molecules, resulting in better models and less biased analysis. Checking the datasets for duplicates identified 55 (44 solutes and 11

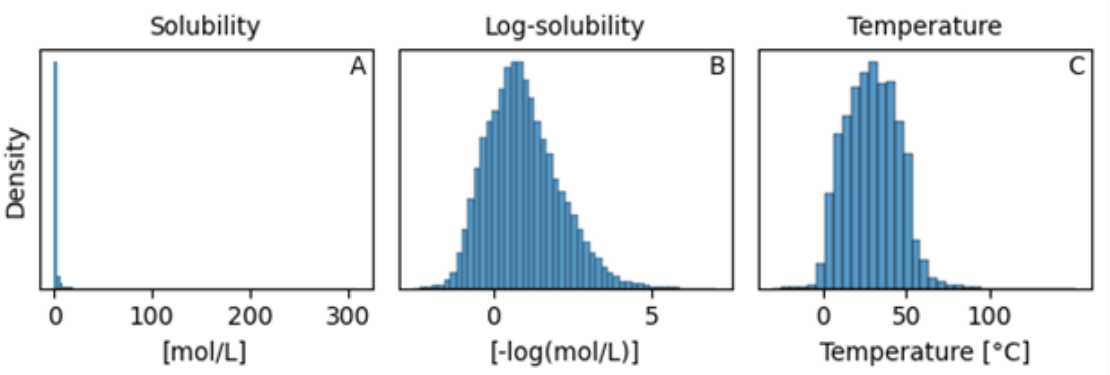

Figure 5.1. Histograms for A) solubility [mol/L], B) log-solubility [-log(S)], and C) temperature after outlier removal.

solvents) duplicates, shared between BigSolDB and QM9, which were removed from BigSolDB. After removing the duplicate entries between the four datasets, ZINC, QM9, BigSolDB, and Eval, evaluating the number of duplicates within each dataset rather than between them. ZINC, QM9, and Eval have no duplicates within them, whereas BigSolDB has duplicate molecules. Duplicate molecules in BigSolDB arise because individual solutes can be paired with different solvents and measured at different temperatures. These repeated combinations are valuable because they capture condition-dependent variation in solubility, including how the same solution behaves across temperatures and how different solutes and solvents interact, while also allowing for a larger dataset. The total and unique molecule counts for each dataset are shown in **Table 5.1**.

To determine which VAE is optimal for generating latent vectors for BigSolDB-solutes and BigSolDB-solvents, the complexity of ZINC and QM9 is compared. The complexity of each dataset is measured using the Bertz complexity index (BertzCT).[359] BertzCT is quantified from heavy-atom graph connectivity and depends on both the number of heavy atoms and how they are connected, as described by atomic degrees. Straight chains or single rings result in low BertzCT values, while molecules with many branches or multiple or fused rings provide higher BertzCT values because their connection patterns are more complex. The BertzCT equation is shown in equation 5.1. The BertzCT value for each molecule is computedand report the average BertzCT across all molecules in each dataset, as shown in **Table 5.1**. Three examples: glycerol, 2,2,6,6-

Table 5.1. Dataset size (After removing incompatible molecules), total and unique molecule count, average molecule size, and average BertzCT of the five datasets (ZINC, QM9, Merged, BigSolDB, and Eval).

Dataset size (Unique)	Dataset size (Unique)	Average molecule size	Average BertzCT
249,455 (249,455)	249,455 (249,455)	23.15	21.66
133,882 (133,882)	133,882 (133,882)	8.80	8.44
383,337 (383,337)	383,337 (383,337)	18.14	17.04
83,099 (1,221)	83,099 (1,221)	16.97	15.44
83,099 (63)	83,099 (63)	4.63	2.18
21 (21)	21 (21)	20.95	19.48
21 (3)	21 (3)	3.29	1.02

$$BertzCT = \ln\left(\prod_{i=1}^{N} d_i!\right) \tag{5.1}$$

tetramethyl-1-piperidinyloxy (TEMPO), and methane, shown in **Figure 5.2**. Glycerol has a BertzCT value of 3.178, TEMPO has a value of 7.049, and methane has a value of 0.

The total number of unique molecules and the average molecule size using the number of non-hydrogen atoms are shown in **Table 5.1**. ZINC exhibits the highest average BertzCT of 21.66, followed by Eval-solutes with 19.48. QM9's average BertzCT is 8.44, with Merged being proportional to ZINC and QM9. BigSolDB-solutes have an average BertzCT of 15.44, being closest to Merged. The solvents have the lowest average BertzCT values: 2.18 for the BigSolDB-solvents and 1.02 for the Eval-solvents (in mostly acetonitrile), approximately four and eight times lower than the next-smallest dataset (QM9), respectively. Based on the average BertzCTs and average molecular sizes of each dataset, it is observed that BigSolDB-solutes are most closely aligned with the Merged dataset, and BigSolDB-solvents with the QM9 dataset. Another observation is that the average molecule size is closely related to the average BertzCT except for the solvents, whose average molecule size is approximately twice as large for BigSolDB-solvents (2.18 average BertzCT versus 4.63 average molecule size) and approximately three times larger for Eval-solvents (1.02 average BertzCT versus 3.29 average molecule size). These trends

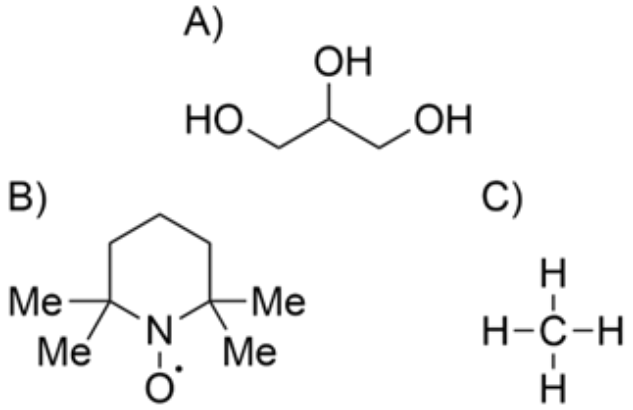

Figure 5.2. A) glycerol, B) 2,2,6,6-tetramethyl-1-piperidinyloxy (TEMPO) C) methane, created using ChemDraw.

are used alongside the later VAE training and PCA evaluation results to determine which VAE is best suited for encoding BigSolDB-solutes and solvents for XGBoost training.

After evaluating the average BertzCTs and average molecular size for each dataset, the final data representations are prepared for XGBoost solubility prediction by training the three VAEs (ZINC-VAE, QM9-VAE, Merged-VAE). Data preparation begins with SMILES strings from the QM9, ZINC, and BigSolDB datasets. However, SMILES strings for the Eval dataset are not included and must be generated from the molecular structures in table 2 of Ahn et al. using ChemDraw.[8,198,199,214] From these SMILES strings, adjacency matrices and node and edge feature vectors are generated, which together represent the molecule in a form that is both compatible and useful for VAEs. The adjacency matrices are binary $n \times n$ matrices, where n is the number of heavy atoms, describing which atom pairs are connected. Node feature vectors encode atomic attributes such as atomic number and charge. In contrast, edge feature vectors encode bond type and related connectivity information; these representations are used to train the three VAEs.

After preparing the data for training the three VAEs, the latent vectors for each molecule are generated using the VAEs. The VAEs in the CGVAE framework are trained similarly to other NNs, with the key differences being that the inputs and outputs are identical and that the decoder applies a specialized reconstruction procedure rather than an NN-based model, as described in the models section. The latent embeddings learned by VAEs encode implicit molecular structures, including atom connectivity, local

substructures, ring systems, and functional group patterns. However, the individual molecular information contained in the latent vectors is arbitrary: they arise from training constrained only by chemical rules and do not map cleanly to interpretable descriptors or linear combinations thereof. As a result, it is not possible to assign descriptors or a linear combination of descriptors to any particular latent vector index. Additionally, information stored in each latent vector index may differ substantially between the three VAEs. However, because reconstruction accuracy can be directly evaluated by comparing the original and generated SMILES, the ability of the VAE to regenerate the input molecules indicates that the latent space captures chemically meaningful information about molecular structure. The VAE results lead to the central theory of this chapter for using VAE latent vectors as descriptors. If the decoder can reconstruct the original molecule from its latent vector, then that vector must contain sufficient information about the molecule to serve as a useful descriptor for predicting molecular properties. The latent vectors used for the final data consist only of the latent distribution means. The latent vector-based approach yields two advantages: 1) the VAEs and XGBoost models can each have their own, more optimized normal variance, and 2) the variance chosen for the XGBoost model can be adjusted without needing to retrain each VAE, which is far more time-consuming compared to training an XGBoost model. The key advantage of using latent vectors as descriptors is that they are implicit and continuous, allowing them to fill gaps between the explicit molecular descriptors generated by RDKit and PaDEL.

The latent vectors for the BigSolDB-solutes and the BigSolDB-solvents derived from BigSolDB are augmented with the RDKit and PaDEL molecular descriptors shown in **Table 5.2**. Different descriptors are used for the solutes and solvents, enabling further optimization of each. The data is further augmented with 256-bit ECFPs generated using RDKit. The ECFPs encode substructures within molecules into fixed-length binary vectors. In this representation, neighborhoods around each atom are hashed into bits.

Molecules sharing similar local environments exhibit more overlapping fingerprints, providing a complementary view of molecular structure to both descriptors and latent vectors. The process for generating ECFPs is detailed in Rogers et al.[360] Augmenting the latent vectors with traditional descriptors and ECFPs leverages descriptors known to correlate with solubility while allowing the learned representations to supply broader structural information that can refine the performance of the models. After processing the data to include latent vectors, RDKit and PaDEL descriptors, ECFPs, and temperature, leaving 639 descriptors.

The complete pipeline from dataset processing to solubility predictions is summarized in **Figure 5.3**; the top panel illustrates VAE training, BO, graph encoding, and latent-space generation; the middle panel shows data preparation, including construction of latent vectors and all descriptor sets (RDKit, PaDEL, ECFPs, and temperature); and the bottom panel depicts XGBoost training using this combined feature set to predict solubility.[215,217] Having used the three VAEs to generate latent vectors augmented with RDKit and PaDEL molecular descriptors, ECFPs, and temperature, each descriptor's

Table 5.2. Data descriptors used from RDKit and PaDEL.

Solutes	Solvents
Molecular weight (MolWt)	Molecular weight (MolWt)
Partition Coefficient (MolLogP)	Partition Coefficient (MolLogP)
Topological polar surface area (TPSA	Topological polar surface area (TPSA
Hydrogen-bond donors (nHBDon)	Hydrogen-bond donors (nHBDon)
Hydrogen-bond acceptors (nHBAcc)	Hydrogen-bond acceptors (nHBAcc)
Number of rotatable bonds (NumRotBonds)	Fraction of sp^3 carbons (FracCSP3)
Fraction of sp^3 carbons (FracCSP3)	XLogP
Heavy atom count (HeavyAtomCount)	ALogP
Ring count (RingCount)	
XLogP	
ALogP	

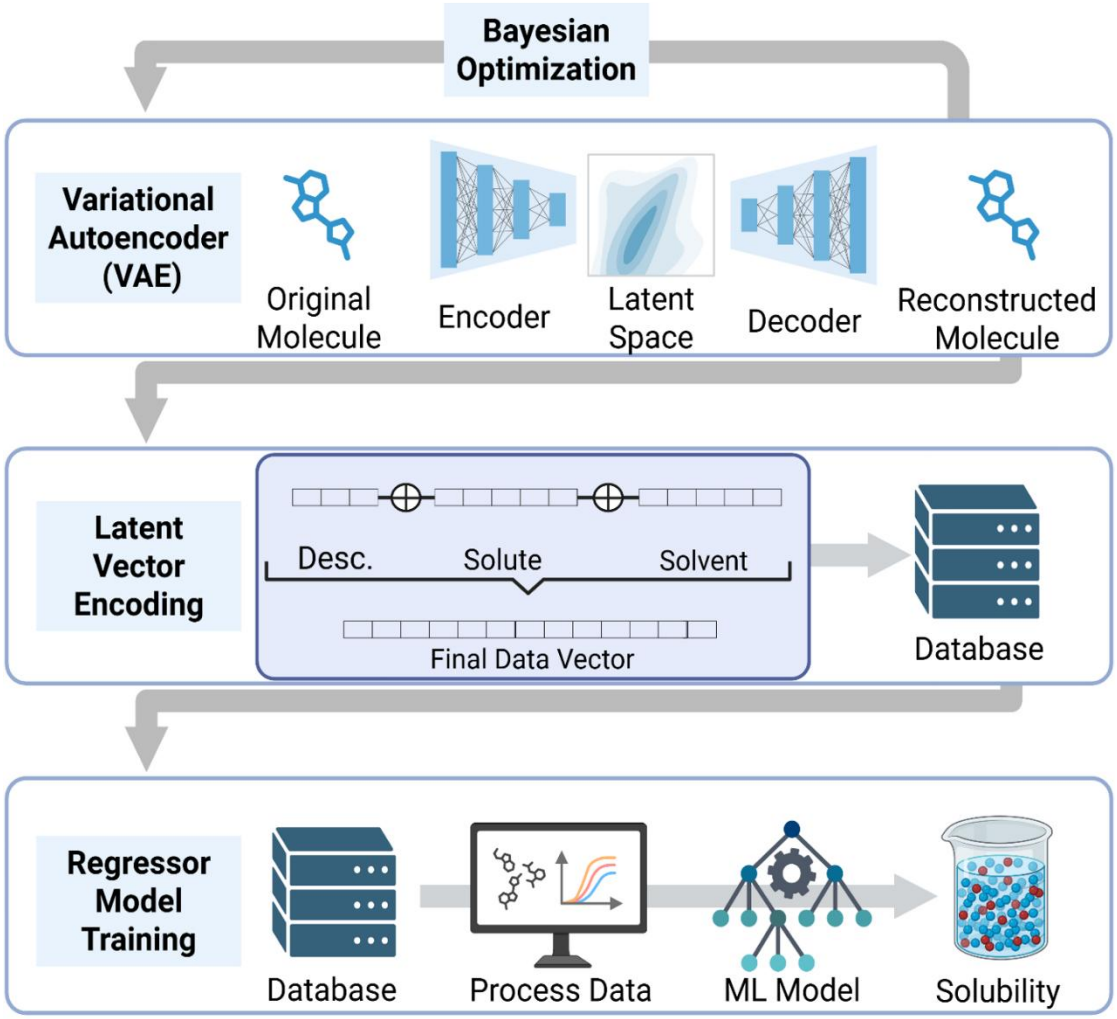

Figure 5.3. ML workflow to predict solubility using VAEs and BO (top) to generate an implicit representation (latent vectors) of the solutes, solvents, and temperature (middle). Then, using the generated representations, train an XGBoost model to predict solubility (bottom).

influence on model outputs is evaluated using SHAP to quantify how much each feature increases or decreases the SHAP explainer's prediction, and then takes the absolute value. In this case, a tree-based surrogate model, which is most comparable to XGBoost, is used to generate the SHAP values. SHAP assigns a value to each descriptor for each data point in BigSolDB after data generation and processing. In the SHAP summary plot shown in **Figure 5.4**, descriptors are ranked vertically by their mean absolute SHAP values; those at the top have the greatest average influence on the predictions of the SHAP explainer model.[361] The top-ranked descriptors are the solute ECFPs, solute molecular weight, temperature, the solute latent vectors, the solvent ECFPs, and the solvent latent vectors. The horizontal axis represents the SHAP value for an individual datapoint. The solute ECFPs, solute molecular weight, solute topological polar surface area (TPSA), and variants of LogP for solutes and solvents show a broader horizontal spread of SHAP values compared to the other descriptors. The horizontal spread indicates that the effect of these descriptors on a prediction varies more across data points. In contrast, temperature, the solute latent vectors, and the solvent ECFPs have high mean SHAP values; they have a more consistent effect on the SHAP explainer model outputs, as indicated by their narrower horizontal spread. Finally, the remaining RDKit and PaDEL descriptors, especially solute H-bond donors and acceptors, and most of the solvent descriptors appear lower on the plot. They are tightly concentrated around a SHAP value of zero, meaning they produce only small, infrequent changes in the output. The color gradients further show how the descriptors within an individual data point shift the prediction: for example,

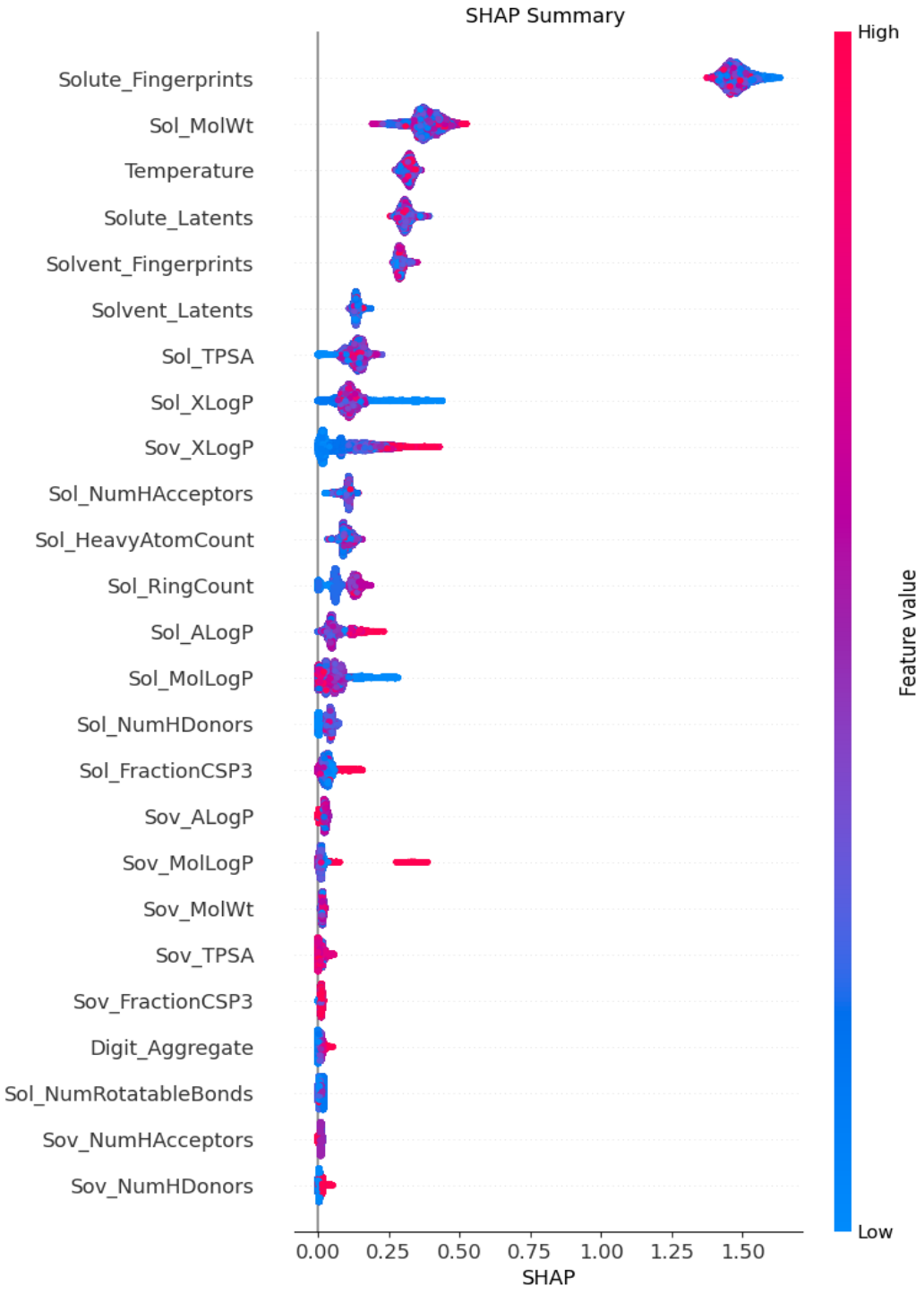

Figure 5.4. SHAP values (red-blue gradient) of all model features with each set of latent values and molecular fingerprints aggregated.

clusters of high-value (red) points along the axis indicate descriptors that increase the predicted solubility, whereas low-value (blue) points indicate descriptors that decrease it. The same descriptors can increase (red) or decrease (blue) the output for different individual data points, as shown by overlapping red and blue points at similar SHAP values for the same descriptor. The overlap of high and low SHAP values indicates that the effect of a given descriptor on XGBoost's prediction is strongly nonlinear, rather than following a simple linear trend. The SHAP results in **Figure 5.4** show that solute descriptors are more impactful than solvent descriptors, with the SHAP explainer model depending primarily on solute molecular weight, solute and solvent ECFPs, temperature, and both latent vectors. In contrast, most other descriptors provide only limited finetuning of the predictions.

Model evaluation uses two metric groups, one for the VAEs and one for XGBoost. The VAEs track node loss, edge loss, and KL divergence, and the weighted combination of these three, with the Tanimoto (Jaccard) similarity being calculated after training. Node and edge losses quantify how well the VAEs reconstruct nodes and edges, respectively, while the KL divergence serves as a regularization term. Tanimoto similarity, which compares the ECFPs of the original and generated molecules, can be calculated as shown in equation 5.2:

$$Tanimoto(A,B) = \frac{|A \cap B|}{|A| + |B| - |A \cap B|} \tag{5.2}$$

where |A| is the bit set in the ECFP of the original molecule, |B| is the bit set in the ECFP of the generated molecule, and $|A \cap B|$ is the number of bits in both molecules A and B. The Tanimoto similarity value ranges from zero (no similarity) to one (identical), but it does not scale linearly. The XGBoost model uses R^2 to measure model fit, MAE to measure error, and RMSE to measure precision. Together, these metrics quantify both VAE reconstruction quality and the XGBoost model predictions.

Results and Discussion

To determine a near-optimal latent vector size, BO was used to approximate the optimum latent vector size by evaluating $n + m$ distinct latent vector sizes within the latent-dimension range of 25-100. Only the latent vector sizes are optimized rather than the entire VAE model architecture to keep the search space constrained and to save time. The BO focuses on optimizing the most important aspect of the VAE model, since the latent vectors are inputs to both the VAE decoder and the XGBoost model. Rather than exhaustively testing every possible latent vector size, this approach tests only a fraction of the range using an acquisition function (LCB) to select the optimal latent vector size, substantially reducing runtime while still yielding a good approximation. To visualize the BO results, VAE testing losses of each evaluation were plotted versus the latent vector size,

and the best points (yellow stars). The BO curves of all three VAEs (ZINC-VAE, QM9-VAE, and Merged-VAE) shown in **Figure 5.5** exhibit a similar trend; testing loss remains relatively flat with modest increases at higher latent vector sizes. For all three VAEs, the test loss is effectively constant in a range of latent vector sizes of (25-66 for ZINC-VAE, 25-67 for QM9-VAE, and 25-65 for Merged-VAE), changing by only 2.3% (ZINC-VAE), 2.9% (QM9-VAE), and 3.7% (Merged-VAE), with only a modest increase as the latent vector size approaches 100. The BO evaluated only 10 latent vector sizes, compared with the 75 used in an exhaustive search. These points are concentrated mostly within the lower-loss region due to LCB's balance of exploration and exploitation; as a result, the true upper end of the flat-loss region may be larger. However, determining the full range of the flat loss region would require an exhaustive search, which contradicts BO's advantages. The small variations within the flat loss region are likely attributable to stochastic variations in the gradients of the VAE model, which cause stochastic gradient descent to converge to a different, although similar, local minima. Another factor is that the variance in the latent distribution produced slightly different decoder inputs for the same molecule. Neither of these factors is considered a true performance difference because they arise from random variability rather than a change in the VAE's true accuracy. For higher latent vector sizes, the VAE's test loss changed modestly, by 4.9% for ZINC-VAE, 2.6% for QM9-VAE, and 5.7% for Merged-VAE, with the ZINC-VAE and Merged-VAE showing slightly higher losses at the largest latent vector sizes. In contrast, the QM9-VAE BO curve remained essentially flat across the entire range.

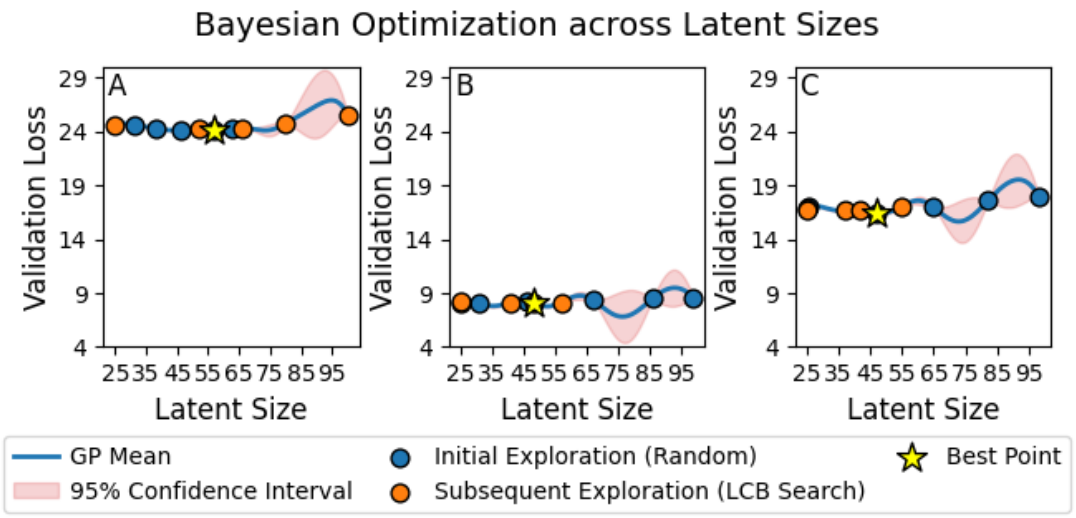

Figure 5.5. Bayesian optimization results for all three models A) the ZINC-VAE, B) the QM9-VAE, and C) the Merged-VAE, with GP means (Blue), 95% confidence intervals (Red), five initial points (Blue), five subsequent points (Orange), and the optimal point (Yellow).

The optimal latent vector size was determined by performing ten BO evaluations as described in the methods (n = 5 pseudo-random initializations and m = 5 subsequent runs using LCB), yielding the best-performing latent vector size for each dataset, indicated by yellow stars. Using BO, the best-performing latent vector sizes of 57 for the ZINC-VAE, 48 for the QM9-VAE, and 47 for the Merged-VAE were identified. These values lie within the relatively flat region of the range. However, given this relatively flat loss landscape in the lower range, the choice of latent vector size appears to have limited impact. The flat loss landscape indicates that selecting a reasonable latent vector size within the lower range is sufficient and that extensive BO is unnecessary. After determining the optimal latent vector size for each VAE (57 for ZINC-VAE, 48 for QM9-VAE, and 47 for Merged-VAE) using BO, shown in **Table 5.3**, each VAE was benchmarked on their respective training datasets (ZINC, QM9, and Merged) shown in **Table 5.3**. The ZINC-VAE exhibited the highest losses across all training metrics (node loss, edge loss, KL loss, and total loss) and the lowest graph similarity. In contrast, the QM9-VAE achieved the lowest losses and highest graph similarity. Despite being trained on the combined ZINC and QM9 datasets, the Merged VAE does not show loss metrics proportional to the dataset sizes of ZINC and QM9, suggesting that increased chemical complexity limits reconstruction quality and graph similarity. In contrast, the BertzCT values and average molecular sizes scale more proportionally with dataset size because they are computed directly from the molecular structures rather than measured by the

Table 5.3. VAE testing results for the solute and solvent datasets with their BO hidden size, node loss, edge loss, KL loss, total loss, and graph similarity.

Dataset	Split	Hidden Size	Node Loss	Edge Loss	KL Loss	Total Loss	Graph Similarity
ZINC	Training	57	0.274	9.700	133.530	23.300	0.304
	Testing		0.278	9.649	132.968	23.340	0.268
QM9	Training	48	0.103	3.493	40.546	7.640	0.491
	Testing		0.097	3.490	40.491	7.625	0.377
Merged	Training	47	0.249	9.250	104.342	19.934	0.360
	Testing		0.230	8.092	91.764	17.427	0.303

Merged-VAE model. Since all three VAEs share the same model architecture, except for latent vector size, these results indicate that model performance is driven more by the molecular structural complexity of each dataset than by model architecture or training set size.

To compare the generated latent vectors across datasets, each VAE was further evaluated using PCA trained on four datasets: ZINC, QM9, BigSolDB-solutes, and BigSolDB-solvents, except for the QM9-VAE, which included only QM9 and BigSolDB-solvents. For each VAE model, the latent vectors were projected onto a two-dimensional space using PCA to transform them, and the resulting distributions for each dataset are shown in **Figure 5.6**. Three PCA plots of latent vectors generated for the three VAEs shown in **Figure 5.6**. Each PCA plot contains points for all datasets compatible with the respective VAE; all six datasets were used for the ZINC-VAE and Merged-VAE PCA plots. QM9, BigSolDB-solvents, and Eval-solvents for the QM9-VAE PCA plot. The clusters for the ZINC-VAE (**Figure 5.6A**) show that the BigSolDB-solute cluster largely overlaps with the ZINC cluster; however, the BigSolDB-solute cluster extends beyond the main ZINC region on the right side and also exhibits a tail at the top of the plot. QM9's latent space cluster is more tightly clustered than that of ZINC and the BigSolDB-solutes, and is contained entirely within the ZINC cluster. The BigSolDB-solvent cluster lies almost entirely within the QM9 cluster, except for a few data points at the end of a tail-like extension. The Eval-solutes lie mainly within the cluster for BigSolDB-solutes, although they are more prominent on the left side, and several points near the top are on the edge.

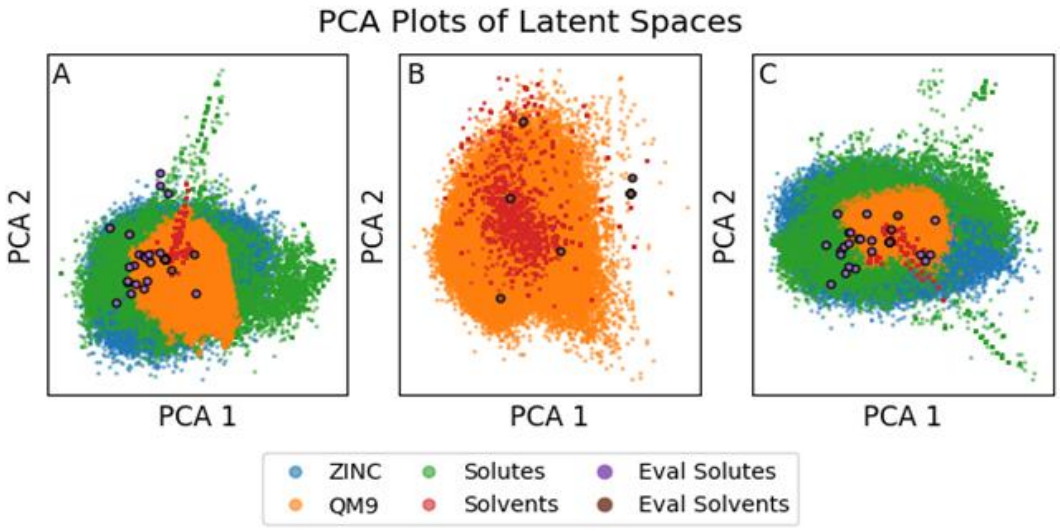

Figure 5.6. Principal component analysis (PCA) for the latent vectors of the three models trained A) (ZINC-VAE), B) (QM9-VAE), and C) (Merged-VAE). Labels for ZINC (Blue), QM9 (Orange), BigSolDB-solutes (Green), BigSolDB-solvents (Red), Eval-solutes (Purple), and Eval-solvents (Brown).

Finally, the datapoints for the Eval-solvents directly correspond to those in the BigSolDB-solvents, meaning these datapoints are shared between the two datasets, resulting in duplicate points. As such, the Eval-solvents do not need to be evaluated separately for any of the PCA plots. The tightness of each cluster is consistent with their BertzCT; higher complexity and larger average molecule size result in a wider-spread cluster, and vice versa.

Since the QM9-VAE (**Figure 5.6B**) uses the QM9 notation to generate data, only QM9, BigSolDB-solvents, and Eval-solvents are evaluated. For the QM9-VAE, the BigSolDB-solvents cluster remains within the QM9 cluster, except for a small number of points near the top edge, although none are clearly outside it. However, the BigSolDB-solvents cluster is less tightly clustered than the QM9 cluster in the PCA plot for the ZINC-VAE, indicating that the QM9-VAE better captures variance in smaller molecules because the scale is not affected by the presence of larger molecules in the ZINC and BigSolDB-solutes datasets.

The PCA plot of the Merged-VAE (**Figure 5.6C**) closely resembles that of the ZINC-VAE; the overall cluster shapes are similar, although appear rotated or flipped, with the BigSolDB-solutes and BigSolDB-solvents clusters showing the greatest visual similarity. The BigSolDB-solutes cluster has several points near the top, outside the main ZINC cluster, which were not present in the ZINC-VAE PCA plot. The QM9 cluster appears more tightly clustered, with the BigSolDB-solvents exhibiting the same behavior. The BigSolDB-solutes and BigSolDB-solvents remain elongated with tail-like regions

extending beyond the cores of the ZINC and QM9 clusters, respectively. In the Merged-VAE latent space, the Eval-solutes lie predominantly within the region occupied by ZINC molecules, likely because the Merged training set includes both larger molecules from ZINC and smaller molecules from QM9. At the same time, the Eval-solutes are more complex and therefore more similar to the ZINC subset relative to the entire Merged dataset. As a result, the Eval-solutes cluster closer to ZINC molecules than to QM9 in the learned embedding space, compared to the ZINC-VAE, where the scaling is unaffected by the smaller molecules from QM9.

These PCA projections reveal how different training datasets affect the learned latent spaces generated by the VAEs, highlighting each VAE's ability to generalize across chemical datasets, identify structural similarities, and indicate potential limits of extrapolation beyond its training data. However, due to compression, some information may be lost, although the PCA transform is performed to minimize (not completely prevent) information loss.

After combining the latent vectors of the BigSolDB-solutes and BigSolDB-solvents with RDKit and PaDEL descriptors, ECFPs, and temperature, three additional PCA plots were generated using the complete descriptors rather than the latent vector-only PCA plots that were used in **Figure 5.5**. The PCA plots in **Figure 5.6** are colored by solvent species.

The PCA visualization of the final data forms several nearly vertical lines, corresponding to distinct solvent species, as shown in **Figure 5.7**, suggesting that solvents play a major role in defining the principal components in contrast to SHAP. The

variation observed both along and perpendicular to these lines reflects differences in solutes, temperature, and stochastic noise. Additionally, evaluating solute-specific effects in detail is impractical given the large number of solutes (1,221), which would require an unmanageable number of distinct colors. Even with this limitation, the PCA plots still captured meaningful structural relationships present in the full, uncompressed dataset. The PCA plot in **Figure 5.6** provided useful analysis compared with the SHAP analysis in **Figure 5.4**, demonstrating the effects of the solutes, solvents, and temperature.

Table 5.4 reports the losses of each VAE model when evaluated using the BigSolDB-solutes and BigSolDB-solvents datasets, using only a single instance of each molecule. The VAE losses evaluated using the BigSolDB-solutes and BigSolDB-solvents are the most important metric to determine which VAE is best suited to generate the latent vectors, in conjunction with the BertzCT and average molecule size comparison in **Table 5.1**, and PCA plots shown in **Figure 5.6** and **Figure 5.7**. For BigSolDB solutes, the ZINC-VAE shows a marginally lower total loss than the Merged-VAE (16.103 vs 16.833) and higher graph similarity (0.177 vs 0.144) for the ZINC-VAE and Merged-VAE, respectively. Because the difference in total loss is low enough to be due to stochastic variance in a similar manner to that of the BO evaluations, other determining factors demonstrate that the ZINC-VAE has a comparable total loss when using the BigSolDB-solutes, a higher graph similarity, and ZINC has a more comparable average BertzCT and average molecule size to the BigSolDB-solutes, and more well-aligned PCA plots in **Figure 5.6**. Further comparing the ZINC-VAE and Merged-VAE evaluated using the

Table 5.4. VAE testing results for the solute and solvent datasets with their BO hidden size, node loss, edge loss, KL loss, total loss, and graph similarity, scaled to account for duplicate molecules within BigSolDB. The best testing results are bolded.

Training Dataset	Evaluation Dataset		Hidden Size	Node Loss	Edge Loss	KL Loss	Total Loss	Graph Similarity
ZINC	Solute	Training	57	0.249	8.800	90.013	18.049	0.227
		Testing		0.592	7.522	82.863	16.400	0.177
	Solvent	Training		**0.024**	1.774	27.692	4.567	0.291
		Testing		**0.018**	1.137	27.348	3.890	**0.342**
QM9	Solute	Training	48	-	-	-	-	-
		Testing		-	-	-	-	-
	Solvent	Training		0.030	**1.327**	**19.347**	**3.291**	0.227
		Testing		0.162	**0.521**	**20.049**	**2.687**	0.177
Merged	Solute	Training	47	0.290	8.870	88.466	18.006	0.193
		Testing		0.282	7.838	83.559	16.476	0.144
	Solvent	Training		0.034	1.563	25.262	4.124	**0.325**
		Testing		0.546	2.790	25.300	3.110	0.192

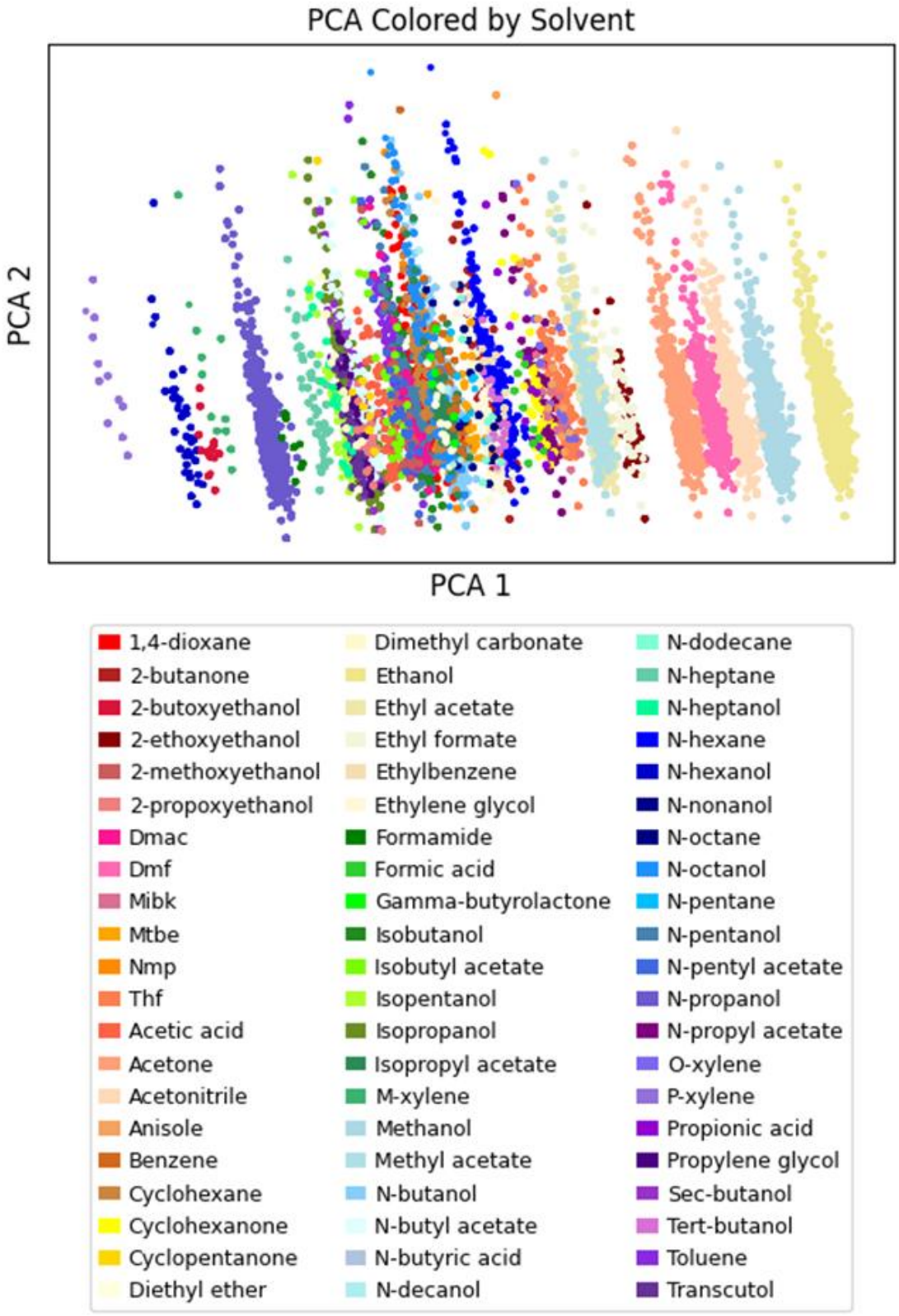

Figure 5.7. Principal component analysis color coded by solvent species.

BigSolDB-solutes, the latent vector sizes differ only slightly between these VAE models (57 for the ZINC-VAE and 47 for the Merged-VAE). The results for the BigSolDB dataset are consistent with the BO analysis: latent vector sizes within a reasonable range had little effect on the VAE results. Instead, the choice is guided by small but consistent advantages in total loss, graph similarity, BertzCT, average molecule size, and the PCA plots, rather than by latent vector size. Therefore, the ZINC-VAE was selected to generate the solute latent vectors. For the BigSolDB-solvents, the QM9-VAE performs best, yielding the lowest total, edge, and KL losses, and has the most similar BertzCT and the average molecule size. Although other VAEs achieve marginally better node loss and graph similarity, they treat total loss as the main indicator of latent vector viability. Finally, the scaling of the other VAEs is adversely affected by the larger ZINC molecules compared to the QM9-VAE. These results make it clear that the QM9-VAE provides the most suitable latent representation for solvent molecules, although the other two VAEs (ZINC-VAE and Merged-VAE) are still viable options.

Now that the final data has been generated and analyzed, it can be used to predict solubility. An XGBoost model was trained to predict solubility using the final set of descriptors (solute and solvent latent vectors, RDKit and PaDEL descriptors, ECFPs, and temperature). Outlier solubility values identified using a modified Z-score were removed (1.89%), and the data were normalized before training.[358] The training results are summarized in **Table 5.5**, and the regression performance and error distributions are shown in **Figure 5.8**. Results from each group K-fold CV test set are used. To better demonstrate

Table 5.5. XGBoost training and testing results [-log(S)], in MAE, RMSE, and R^2.

Dataset	MAE	RMSE	R^2
Training	0.162	0.244	0.954
Testing	0.210	0.309	0.925

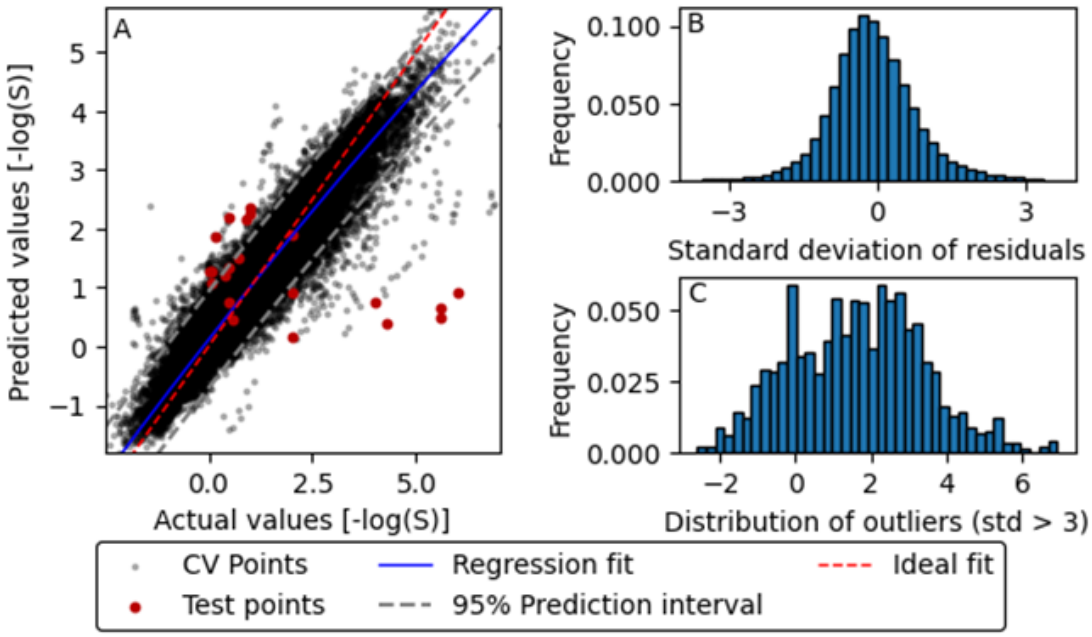

Figure 5.8. A) Regression plots for solubility predictions with the CV testing sets (Black), testing set (Red), the true prediction line (Red), the fit line (Blue), and the 95% prediction interval (Grey), B) distribution of standard deviation of errors (residuals), and C) distribution of predictions with a standard deviation greater than three as a function of the true solubility [-log(S)].

the distribution of errors, the CV test points are partially transparent, revealing the density of the predictions in **Figure 5.8A**. The overall distribution of outliers in standard deviations is shown in **Figure 5.8B**, and the distribution of outliers above three standard deviations, plotted as a function of solubility in $-log(S)$, is shown in **Figure 5.8**. A closer examination of the predictions in **Figure 5.8A** highlights several aspects of XGBoost's performance. Most points cluster tightly around the ideal prediction line, indicating strong agreement between predicted and actual solubility values, as shown in **Figure 5.8A**. Although a relatively large number of outliers are present (1.45% of observations have an error greater than three standard deviations; see **Figure 5.8B**, which shows only small tails), under an ideal normal distribution, only about 0.3% of data points are expected to lie beyond three standard deviations. The increase in outliers suggests that the underlying distribution deviates from normality, as confirmed in **Figure 5.8C**. Despite the high number of outlier errors, their influence appears limited, as the overall distribution of errors shown in **Figure 5.8B** remains narrow and resembles the solubility distribution shown in **Figure 5.1**, both of which are approximately normal with a mean of approximately two.

The XGBoost model demonstrated an error 51.2% lower than the existing state-of-the-art models (0.210 MAE vs 0.43-0.68, both in Log(S)) and 63.5% lower MAE than the GSE, shown in **Table 5.6**.[103,337,343,347,349,350,362] It is observed that solutions with lower solubility values tend to be over-predicted. In comparison, those with higher solubility values are slightly under-predicted; in other words, models generate fit lines with a steeper slope, a trend that is consistent with the studies shown in **Table 5.6**. The bias is present to

Table 5.6. Comparison of ML model results using MAE, RMSE, and R^2 using testing set or equivalent using log(S), with the best results bolded.

Method	Dataset Size	MAE	RMSE	R^2	Paper
Our Work	**83,099**	**0.210**	**0.308**	0.925‡	Our Work
MOE Descriptors	287	0.657	0.936	0.68	Hughes et al.*[103]
HQSAR	257	0.59	-	-	Klamt et al.*[343]
gpHSP	193	0.68	1.02	0.69	Sanchez-Lengeling et al.[337]
COSMO-RS Descriptors	714	0.59	0.79	0.56	Vassileiou et al.[347]
QSPR Descriptors	1,816	66.3% < 0.7	0.71	**0.93**‡	Boobier et al.†[362]
ECFPs	5,081	0.5968	-	0.4922	Ye & Ouyang.[349]
ECFPs + 2D Descriptors	8,431	0.43	0.64	0.88	Tayyebi et al.*[350]

*Uses aqueous solubility whereas this work in non-aqueous solubility.
†For MAE, percentage of errors below 0.7 log(S) and 1.0 log(S) was recorded, rather than MAE itself.
‡ Due to rounding differences between the R^2 using the approach proposed in this chapter and the R^2 from Boobier et al. may be zero or less than presented.

a lesser extent than in these studies. However, it is still present, suggesting that the descriptor set and XGBoost model generalize better than those in previous studies. Further examination of the results shows that the 95% prediction interval (PI) captures the expected spread of individual observations. The PI is calculated by using the maximum error of the top 95% best predictions, which for the XGBoost model is ±0.5350 log(S). To demonstrate the solubility prediction approach in a real-world setting. A total of 21 solutions were selected from table 2 in Ahn et al., which is referred to as Eval.[8] These were selected because they represent experimentally validated solutions covering a broad range of solubilities and solute-solvent pairs under realistic operating conditions. These realistic operating conditions make the Eval dataset a suitable testbed for evaluating XGBoost model performance in a practical, applicable scenario. Predictions for the solutions are shown in **Figure 5.8A** as larger dark red points, and the exact true and predicted values in both $-log(S)$ and mol/L are shown in **Table 5.7**. It is observed that seven of the 21 solutions have low errors either near or within the 95% PI (±0.5350 log(S)). Much of this error arises because many data points from the Eval dataset are out of distribution for BigSolDB, meaning the molecules in the Eval dataset fall outside the chemical and property space spanned by BigSolDB, particularly in the high-solubility range. The difference in results between the BigSolDB and Eval datasets is because the solutions in the Eval dataset have more complex solutes than those typically found in BigSolDB and include ionic species, whereas BigSolDB entries are primarily neutral molecules.[8,104]

Table 5.7. Predicted vs True solubility values in [mol/L] and [-log(S)] for 21 solutions from table 2 in Ahn et al (Eval-solutes and Eval-solvents). The solution numbers use the same numbering system as the ROM column in table 2 of Ahn et al.[8]

Solution	True Solubility [mol/L]	Predicted Solubility [mol/L]	True Solubility [-log(S)]	Predicted Solubility [-log(S)]
1f	0.1000	0.003130	1.000	2.500
1g	0.1290	0.005500	0.8900	2.260
2a	0.1050	0.004890	0.9800	2.310
4a	0.000003000	0.3290	5.700	0.4830
5a	0.01000	0.1090	2.000	0.9640
5b	0.00005000	0.3430	4.300	0.4640
5c	0.000003000	0.2210	5.600	0.6550
5d	0.3390	0.006670	0.4700	2.180
5e	0.4070	0.06190	0.3900	1.210
7a	0.3160	0.1760	0.5000	0.7540
7b	0.9270	0.04490	0.03300	1.350
8b	0.0001000	0.1400	4.000	0.8550
10a	0.01000	0.01810	2.000	1.740
11f	0.8910	0.05120	0.5000	1.290
13a	0.8710	0.1020	0.6000	0.9900
13b	0.3160	0.05550	0.05000	1.260
15c	0.2510	0.3890	0.06000	0.4100
15d	0.01000	0.5060	2.000	0.296
15f	0.7080	0.02020	0.1500	1.690
17c	0.2000	0.04070	0.7000	1.390
18a	0.000001000	0.1140	6.000	0.9430

Further exacerbated by BigSolDB only containing 1,221 solutes and 63 solvents, and therefore cannot encompass the full chemical space, although BigSolDB is still by far the largest available solubility dataset.

Next, the average loss curves for the CV train and test sets are analyzed, providing further evidence of model stability. Convergence occurred rapidly within the first ~250 iterations, followed by slower but steady improvements through 1,500 iterations, as shown in **Figure 5.9**. Notably, the CV training and test loss curves followed nearly identical trajectories throughout, showing no evidence of overfitting within the chosen iteration range. However, they began to diverge near the end. The close alignment between the two curves reinforces the reliability of the model for both training and unseen test data.

When evaluated against three standard performance metrics (MAE, RMSE, and R^2), this approach consistently outperforms state-of-the-art methods. For XGBoost model training and for comparison with other works, the data is transformed using -log(S), in the original units; mol/L is used for real-world evaluation. To show the results before the log transform in mol/L, each prediction and target value is converted back to the original unit, and MAE and RMSE are recalculated. Evaluating and comparing the results, achieving the lowest MAE (0.210 log(S), 0.490 mol/L) and RMSE (0.308 log(S), 3.212 mol/L). MAE remains relatively small after reversing the log transform. In contrast, RMSE increases more substantially because squaring the residuals amplifies the influence of large errors, making RMSE more sensitive to outliers than MAE. The effect of larger errors is further intensified when predictions are converted back to the

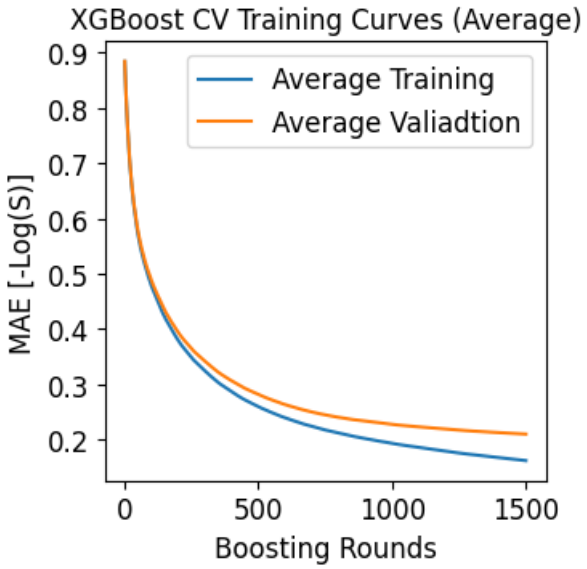

Figure 5.9. Loss curves five-fold cross group-wise testing with XGBoost.

original scale, since any large deviation in -log(S) space becomes significantly larger in mol/L. R^2 is not recalculated because it measures the XGBoost model fit rather than the true values versus predictions. Since the scale between the predictions and targets is no longer the same as the scale the XGBoost model was trained with, measuring R^2 would not be meaningful.

The MAE and RMSE presented in this chapter are substantially lower than reported values in the literature (MAEs ranging from 0.43-0.70 log(S) and RMSEs from 0.64-1.02 log(S)). These improvements represent more than a two-fold reduction in prediction error. In terms of R^2, this approach achieves 0.925, which is among the highest reported and exceeds most prior work, though performance across studies on this metric is more variable (values spanning 0.4922-0.93). A direct comparison is summarized in **Table 5.6**. The work presented in this chapter achieved the strongest predictive performance. Several factors contribute to this improvement. First, the BigSolDB dataset is significantly larger than any prior work; nearly ten times as large as the next largest study, providing a broader and more representative foundation for XGBoost model training. Second, this methodology integrates both implicit and explicit molecular representations of solutes and solvents, with latent vectors augmenting areas not captured by descriptors and fingerprints. Finally, unlike most prior studies that restrict their analysis to either a single solute, a single solvent, or a fixed temperature range, BigSolDB encompasses a wide range of solute-solvent pairs across multiple temperatures. This added variability not only improves generalization but also more closely reflects real-world conditions.

Conclusion

Solubility is a widely studied property both experimentally, computationally, and with ML, although it remains a challenging task for ML to predict. The work in this chapter demonstrates that combining a graph-based VAE with conventional molecular descriptors and an XGBoost (gradient-boosted tree) yields an accurate, generalizable, and transferable framework for predicting non-aqueous molecular solubility across diverse solute-solvent pairs. Leveraging large, chemically broad pretraining sets (QM9 and ZINC) together with an experimental database (BigSolDB), the proposed pipeline reduces prediction error by 51.2% relative to descriptor-only, COSMO-RS-derived, QSPR, and other computational approaches, highlighting the value of learned graph representations for capturing complex solute-solvent interactions in real-world scenarios.

Beyond improving solubility predictions, this representation offers a flexible foundation for data-driven screening and design in application domains where solubility is a performance bottleneck, including electrochemical energy storage, separations, and pharmaceuticals. By decoupling the graph encoders from the downstream regressor, the approach can be readily extended to incorporate task-specific loss functions, uncertainty quantification, or active-learning pipelines that prioritize new measurements in underexplored regions of chemical space.

At the same time, the present study is constrained by the quality and quantity of existing solubility measurements, as well as its focus on single-solute systems. Future work

should incorporate larger, task-specific datasets for electrolytes, multi-solvent systems, and more complex systems, and compare performance against a broader range of ML approaches. Expanding these evaluations across richer chemical spaces and benchmark models will clarify when data-driven solubility prediction is robust enough for practical decision-making in formulation and solvent selection. In parallel, integrating model designs that better reflect underlying chemistry, such as physics-informed NNs, along with individual predictive uncertainties, such as those generated by GP models, can improve both the reliability and transparency of these predictions.

Together, these advances will help transition ML-based solubility models to routinely used tools in molecular design workflows. For μE electrolytes, accurate solubility predictions minimize diluent volume needed for solid oils, directly maximizing energy density while maintaining type IV phase stability. The proposed graph encoder architecture enables transfer learning to new solvents or other multi-component systems.

CHAPTER 6:

MACHINE LEARNING APPROACH FOR DETECTING NANOPARTICLES IN FUEL CELL MATERIALS VIA STEM IMAGING

Abstract

ML offers new opportunities for accelerating materials characterization by automating the extraction of structural information from microscopy data. In this work, a data-driven framework for detecting and characterizing nanoparticles is developed using images generated from multislice STEM simulations. Unlike conventional supervised approaches that rely on manually curated labels, this approach employs an unbiased labeling strategy derived from the underlying ASE model. Labeling the edges in this manner ensures consistent training data and avoids human-introduced biases. Using a Mask R-CNN architecture, the model learns to identify nanoparticle boundaries and morphological features directly from simulated STEM images, enabling accurate nanoparticle segmentation under a variety of imaging conditions, including when there are varying levels of nanoparticle overlap. The model is further validated using real-world STEM images to verify the transferability between multislice and real STEM images. By bridging simulation, ML, and experimentation, this work establishes a foundation for automated nanoparticle characterization and analysis, including the extraction of particle size, shape, spatial distribution, and overlap statistics, enabling high-throughput, quantitative tracking of morphological metrics relevant to device performance.

The work presented in this chapter was done in conjunction with the Center for Nanophase Materials Sciences (CNMS) at ORNL. I thank Dr. Michael Zachman, Dr. Thomas Pfeifer, Dr. Haoran Yu, Dr. David Cullan, and Dr. Lynda Amichi for their support with this project.

Introduction

Nanoparticle degradation in polymer electrolyte membrane fuel cell (PEMFC) electrodes constitutes a primary durability challenge, where platinum (Pt) catalysts undergo sintering, Ostwald ripening, and oxidative dissolution that progressively diminishes electrochemical surface area and power density. Quantifying these spatially heterogeneous degradation mechanisms across entire electrodes is essential for understanding their impact on device performance and lifetime. Yu et al. recently established automated analytical electron microscopy for tracking this evolution, revealing distinct degradation signatures across fresh, conditioned, and failed electrodes. Pt particles grow from ~3-4 nm to >5 nm with larger tails coalescing into irregular aggregates due to Ostwald ripening.[51,363–368] However, their workflow relies on exhaustive semi-automated particle detection across thousands of $200 \times 200nm$ high-angle-annular dark-field (HAADF) scanning transmission electron microscopy (STEM) micrographs ($0.2nm/pixel$), where overlapping nanoparticles, subtle contrast gradients, and beam-induced artifacts limit accurate particle segmentation, particularly for overlaps, and thresholding lacks objective bias assessment for edge placement.[51]

Real-world STEM datasets exacerbate these limitations: sparse ground-truth labels (hours per image), inconsistent acquisition conditions (dose spanning $10 - 10^6$ $\frac{e^-}{Å^2}$, defocus drift, detector geometry), and incomplete boundary delineation for particles overlapping in projection hinder supervised learning deployment.[51,365,369] Physics-based simulation offers

a path forward, but conventional rendering struggles to capture experimental realism at production scales; probe coherence envelopes, thermal diffuse scattering, and Poisson-limited SNR must be faithfully reproduced.[368,370]

Automated nanoparticle detection in HAADF-STEM images remains challenging due to overlapping particles and subtle contrast gradients.[371] Traditional thresholding and watershed methods often mis-segment overlapping nanoparticles, yielding inaccurate size statistics without manual intervention, which limits throughput. [372] High-throughput SEM/EDS approaches classify thousands of environmental particles, but STEM's distinct Z-contrast mechanism precludes direct transfer.[373] Trainable segmentation tools such as ParticleSpy achieve reasonable accuracy with minimal (~100 pixel) labels but falter on subtle particle boundaries where ground-truth delineation is uncertain and human bias cannot be avoided.[372] U-Net CNNs improve HAADF-STEM segmentation for bimetallic Pt/Ni catalysts over classical methods, yet demand real-image labeling prone to human bias.[374] ML frameworks such as YOLOv8 with weighted box fusion enable rapid vesicle detection (~2s/image) but require morphology-specific retraining.[375] Localization U-Nets enable sub-pixel atomic column detection for model systems like Pt/CeO2 interfaces, but real electrodes exhibit disordered carbon supports and heavy nanoparticle overlap.[376] While in situ TEM reveals Ostwald ripening kinetics in Pt/SiO2, ex situ scalability remains essential for PEMFC durability tracking.[366] Recent automated pipelines using DetectNano deliver high-throughput TEM sizing but still rely on curated real datasets vulnerable to acquisition inconsistencies.[375]

Unlike these approaches requiring exhaustive manual labeling or real microscope time, the PySlice simulation generates 2,000 physically-accurate STEM images paired with atomic-hull derived instance masks; after instrument-matched post-processing (partial coherence blurring, Poisson noise, CLAHE), a Mask R-CNN trained solely on these simulations transfers directly to real PEMFC electrodes. This eliminates human bias, handles variable overlaps via multichannel instance masks, and achieves near-identical morphological statistics while outperforming thresholding/watershed on crowded catalysts, providing scalable metrology for degradation studies without ground truth acquisition.

The HAADF-STEM images are simulated from materials generated using the ASE models of gamma-distributed Pt nanoparticles, truncated to mimic realistic distributions better. Multislice propagation at $7mrad$ convergence yields $Z^2 - contrast$ images processed through an instrument-matched cascade: coarse and fine Gaussian blurring simulates partial coherence, Laplacian sharpening enhances particle boundaries, CLAHE boosts local contrast for overlapping regions, and logarithmic Poisson shot spans low-dose beam-sensitive to high-dose reference regimes. Unbiased instance masks derive directly from atomic hulls, natively handling overlaps via multichannel labeling (N particles to N binary mask channels). Unlike typical image augmentation, rotation and flipping are skipped to preserve the crystal face orientations visible in HAADF-STEM images, essential for measuring how Pt nanoparticles form their flat surfaces.[243,244] The trained model works immediately on real fuel cell images from Yu et al. without retraining. The

200×200nm images are tiled into 100 corresponding 20×20nm patches, apply only basic sharpening and contrast enhancement, and get accurate detection, sizing, and shape measurements.[51]

The work in this chapter proposes a PySlice-based framework that eliminates manual labeling and real microscope image acquisition for ML training, replacing exhaustive HAADF-STEM data collection and annotation with physics-accurate simulations that capture experimental imaging variability. The transferability to PEMFC micrograph data from Yu et al., achieving robust nanoparticle detection, sizing, and morphology analysis across degradation states, validates simulation as a complete training data surrogate for production microscopy workflows.[51] Using purely multislice-based computational training data accelerates fundamental understanding of Pt degradation mechanisms while providing metrics to guide development of nanostructured supports for fuel cell electrodes. The proposed simulation-to-deployment paradigm establishes a scalable foundation for materials informatics across electron microscopy challenges.

Using PySlice, a GPU-accelerated multislice electron scattering simulator that generates realistic HAADF-STEM images from atomic coordinates by propagating the electron probe wavefunction through crystal potentials slice-by-slice. It incorporates realistic probe aberrations, annular dark-field detector geometries, and thermal diffuse scattering via frozen phonon configurations, enabling the generation of large-scale datasets with controlled nanoparticle morphologies, ideal for training overlap-robust Mask R-CNN instance segmentation models.

To generate the nanoparticle clusters for the PySlice simulation, the ASE library was used. Using ASE, the atom objects were generated to define the atom type (Pt) and position on the plane, which is 200×200Å. Using ASE's bulk function to generate crystal structures that make up the particles. The particle's length, width, and height are defined using a Gamma distribution shown in equation 6.1. The output of the gamma distribution

$$f(x;\alpha,\theta) = \frac{\beta^{\alpha} x^{\alpha-1} e^{-\beta x}}{\Gamma(\alpha)} \; for\; x > 0, \alpha = 5, and\; \beta = 0.1\text{Å}^{-1} \tag{6.1}$$

is the truncated such that values below 40Å become 40Å and values above 120Å become 120Å, ensuring that every particle is bounded within the plane. The particle's length, width, and height are further scaled such that they are proportional to each other by scaling the height using $height = \max(height_0, 0.5 \cdot min(length, width_0))$ and then the width using width $\max(\text{width}_0, 0.5 \cdot min(length, height))$ ensuring an ellipse-like particle structure. After generating the particle cluster, this process is repeated for N particles using a Gamma distribution shown in equation 6.2.

$$N = \frac{\beta^{\alpha} x^{\alpha-1} e^{-\beta x}}{\Gamma(\alpha)} \; for \; x > 0, \alpha = 3, and \; \beta = 4.5 \tag{6.2}$$

After generating N particles, each particle is then placed and oriented on the plane, repeating pseudo-randomly for each particle, with the placement being constrained such that the entire particle is within the 200×200Å field of view. The particle placement process repeats until proper placement and orientation within the plane are found. If a proper placement and orientation cannot be found after M attempts, the process restarts with N being decremented by one. Improper placement and orientation can be caused by a high number of large particles, which are difficult to fit within the plane in a pseudo-random manner. Keeping the particles fully within the plane ensures that the plane and particles that are used to simulate the HAADF-STEM images are valid by ensuring that the length and width of all particles can be properly measured by being fully within the plane.

STEM image data were generated using a conventional multislice algorithm via the PySlice code package, enabling quantitative simulation of HAADF contrast under realistic experimental conditions without requiring extensive microscope time.[377] The incident STEM probe is first defined in reciprocal space by an aperture-limited plane wave with a 7mrad convergence semi-angle, and the corresponding real-space probe is obtained by inverse Fourier transform.

The sample is discretized into thin slices of thickness $\Delta z = 1nm$ along the beam direction. For each slice, the electron wavefunction experiences a phase shift determined by the projected potential $t(\mathbf{r})$ shown in equation 6.3, followed by free-space Fresnel propagation shown in equation 6.4 to the next slice. This sequence of transmission and propagation operators is applied iteratively until the exit surface is reached, yielding a complex exit wavefield that encodes both coherent and dynamically scattered electrons.

$$t(\mathbf{r}) = e^{i\sigma V(\mathbf{r})} \tag{6.3}$$

In equation 6.3, $\boldsymbol{r}$ is the transverse position vector, σ is the relativistic interaction constant set to $\sigma = 0.1$ in this work, and $V(\boldsymbol{r})$ is obtained from tabulated atomic scattering factors for Pt and C. Between slices, the wavefunction is propagated in reciprocal space using the Fresnel propagator,

$$P(\boldsymbol{k}) = e^{-i\,\pi\,\lambda\,\Delta z\,|\boldsymbol{k}|^2} \tag{6.4}$$

where $\boldsymbol{r}$ is the scattering vector in reciprocal space, $\lambda = 0.0251\text{Å}$ is the electron wavelength at the chosen acceleration voltage ($200\,keV$), and $\Delta z = 1nm$ is the slice thickness. The complete propagation through N slices is shown in equation 6.5.

$$\psi_{n+1}(\boldsymbol{r}) = F^{-1}\{P(\boldsymbol{k}) \cdot F[t(\boldsymbol{r}) \cdot \psi_n(\boldsymbol{r})]\} \tag{6.4}$$

$$\psi_{probe}(\boldsymbol{q}) = A(\boldsymbol{q}) \cdot e^{i \cdot \chi(\boldsymbol{q})} \tag{6.5}$$

In equation 6.4 and 6.5, F and F^{-1} are Fourier and inverse Fourier transforms, $\mathbf{q}$ is the transverse spatial frequency, $A(\mathbf{q})$ is the aperture function that truncates the probe at the 7mrad convergence semi-angle, and $\chi(\mathbf{q})$ encodes the lens aberrations through the microscope contrast transfer function.[378]

For each element Z, the momentum-space form factor $f(\boldsymbol{q})$ is expressed in equation 6.6.

$$f(\boldsymbol{q}) = \sum_{i=1}^{3} [\frac{a_i}{|\boldsymbol{q}|^2 + b_i} + c_i e^{-d_i |\boldsymbol{q}|^2}] \tag{6.6}$$

Where coefficients $a_i b_i c_i d_i$ are shown in Kirkland et al. The projected potential for each slice V($\mathbf{k}$) is shown in equation 6.7, which sums the atomic form factor. fz_j for all atoms j of type z at positions $\boldsymbol{r}_j$ within the slice bounds.

$$V(\boldsymbol{k}) = \sum_j f z_j(\boldsymbol{k}) e^{-2\pi k \cdot \boldsymbol{r}_j} \tag{6.7}$$

The detector response $D(\boldsymbol{k})$, shown in equation 6.8,

$$D(\boldsymbol{k}) = \begin{cases} = 1, k_{Dmin} \leq \boldsymbol{k} \leq k_{Dmax} \\ = 0, else \end{cases} \tag{6.8}$$

after the final slice, the exit wave ψ_{exit} is Fourier transformed to reciprocal space and integrated over the HAADF detector range to obtain the image intensity.

$$I(\boldsymbol{R}_j) = \int_{\theta_{\text{in}}}^{\theta_{\text{out}}} \left|\psi_{exit}(\boldsymbol{k}, \boldsymbol{R}_j)\right|^2 D(\boldsymbol{k}) \cdot dk \tag{6.9}$$

In equation 6.9, θ_{in} and θ_{out} define the inner and outer collection angles, which are 68 mrad and 220 mrad, respectively, $I(R_{\text{j}})$ represents the final value of the image. Mapping

$I(R_j)$ across all probe positions yields HAADF-STEM images with contrast where brightness scales approximately with the square of the atomic number for Pt nanoparticles embedded in a lighter carbon support. The final simulated images are 512×512 pixels at a sampling of 0.39Å per pixel (200×200Å), matching the experimental STEM field of view.

Due to computational constraints, each probe was cropped to $1 \times 1\, nm$ prior to propagation through the potential. Cropping the probe captures the majority of the probe according to the $d_{\frac{1}{e^2}} = \frac{1.22\lambda}{\alpha} = 4.37$Å, and captures all Airy tails above 7% intensity, enabling propagation of a significantly smaller $100 \times 100 - pixel$ waveform (0.1Å sampling) through sub-regions of the potential for each probe position, as opposed to propagating the full $2000 \times 2000 - pixel$ waveform, the majority of which is comprised of zero or near-zero values and contributes negligibly to the final signal. Additionally, only probe positions on and near particles (1nm cutoff distance) were computed, as the STEM signal from all points in vacuum is uniform. An example of the raw multislice generated images is shown in **Figure 6.1**.

After generating the ASE clusters, the coordinates of the center of the outermost atoms on the xy plane are used to create an unbiased particle outline, compared to manually drawing particle edges or using thresholding values from the scan. For each scan with N particles,

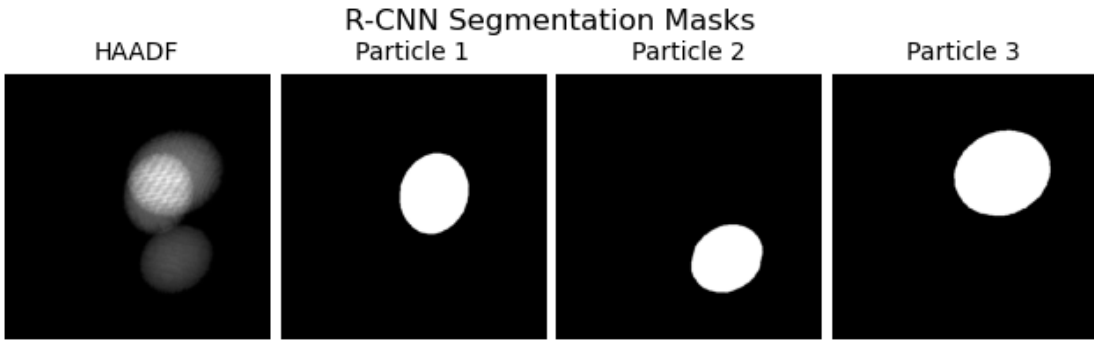

Figure 6.1. Example of generated data (A) HAADF-STEM image generated using PySlice, (B-D) particle masks generated for each particle in the HAADF-STEM image.

a particle mask is created with N channels, one for each particle, with the edges filled, representing a binary particle or background. Because each particle is given its own channel, the model can capture particle overlap by having separate labels for each particle rather than a single label for the entire scan. Additionally, this allows for a variable number of particles to be represented. An example of the generated HAADF-STEM images and label masks is shown in **Figure 6.1**. The masks have the same pixel density as the HAADF-STEM images.

After generating the HAADF-STEM images and particle masks, a series of filters and transforms is applied to better match the real HAADF-STEM data and to improve model performance. First, Gaussian noise is applied, followed by a set of Gaussian blurs: a coarse blur (0.585Å) and a fine blur (0.195Å), both using a uniform distribution. Large-scale aberrations are then added to simulate detector drift and beam instabilities. Then the images are sharpened using Laplacian sharpening to enhance particle edges and to mitigate the negative effects of the Gaussian blur while keeping the positive effects.[379] Then local contrast is boosted using CLAHE to provide better detection of particles that have overlap.[380] Large-scale aberrations are then added to simulate detector drift, beam instabilities, and sample drift commonly observed in real STEM acquisitions. Finally, Poisson (shot) noise is applied with gains of 65.7-6,575 $\frac{e^-}{\text{Å}^2}$ using background-heavy noise modeling with 15-20% gain compared to particles along a uniform distribution, samples of the Poisson noise are shown in **Figure 6.2**. These filters allow the PySlice-generated STEM

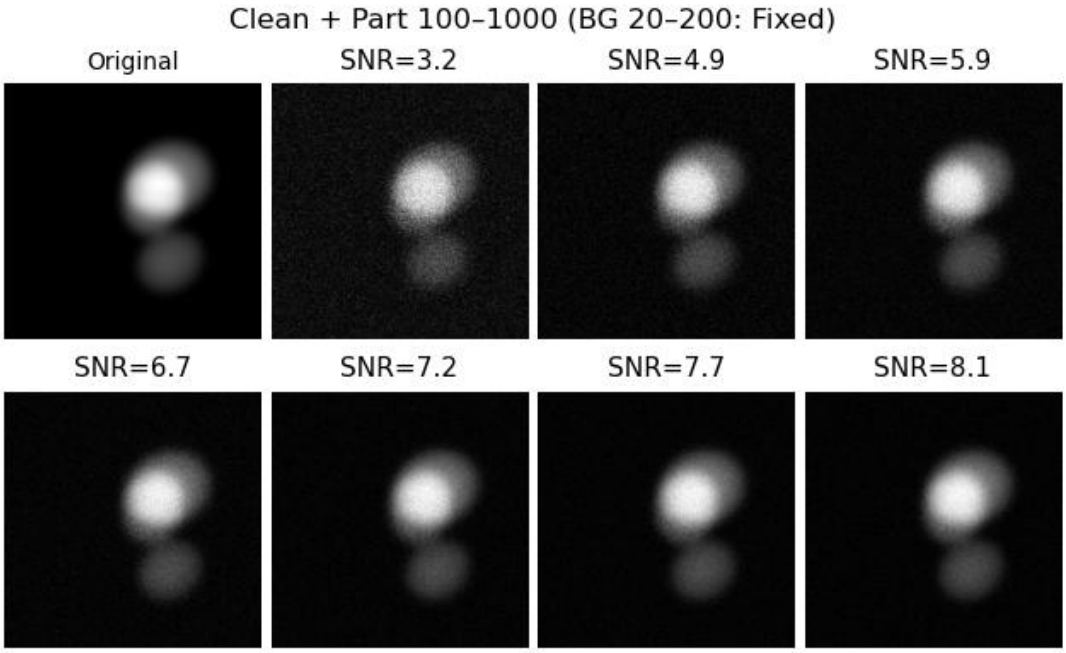

Figure 6.2. Poisson (shot) noise samples along the entire range in SNR.

images to better reflect real HAADF-STEM data. Finally, **Figure 6.3** shows the fully processed data.

To evaluate the PySlice-trained model on real-world data, the model is tested it on samples from Yu et al which is shown in **Figure 6.4**.[51] Since the images from Yu et al. cover $200 \times 200nm$ while the PySlice data spans $200 \times 200\text{Å}$ ($20 \times 20nm$), these images are tiled into 100 non-overlapping tiles ensuring consistent scaling between PySlice data and beginning of test (BOT) data (before degradation) in pixel/Å, the tiling is shown in **Figure 6.5**. The final processed data is shown in **Figure 6.6**. These real HAADF-STEM images provide an unbiased testbed with authentic nanoparticle size, aspect ratio, and count distributions, demonstrating PySlice's ability to generate surrogate training data compatible with production microscopy. Finally, post-processing detected edge particles are removed to ensure that the ferret diameter, area, and aspect ratios are accurate since edge particles are not fully in frame, and thus, the full metrics cannot be determined.

Methods

For the simulated HAADF-STEM dataset, an image segmentation model capable of accurately identifying and separating closely packed nanoparticles that may overlap is needed. Traditional thresholding or segmentation techniques perform poorly in

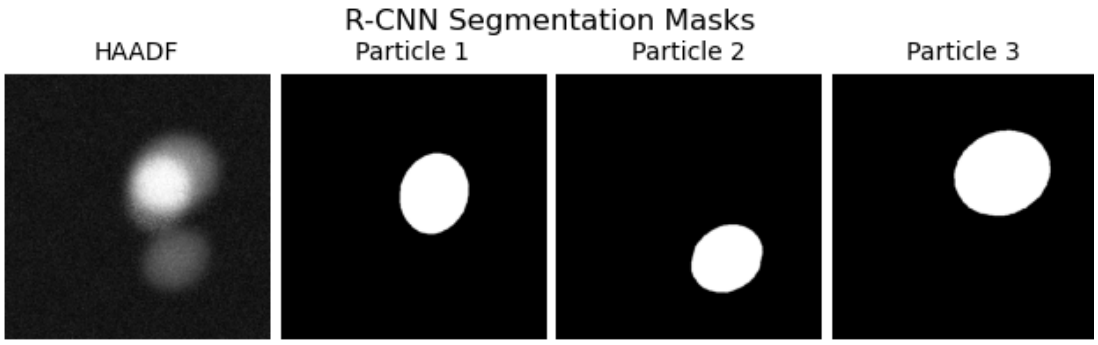

Figure 6.3. Example of processed data (A) HAADF-STEM image generated using PySlice with Gaussian noise, Gaussian blur, Laplacian sharpening, CLAHE, and Poisson noise applied (B-D) particle masks generated for each particle in the HAADF-STEM image.

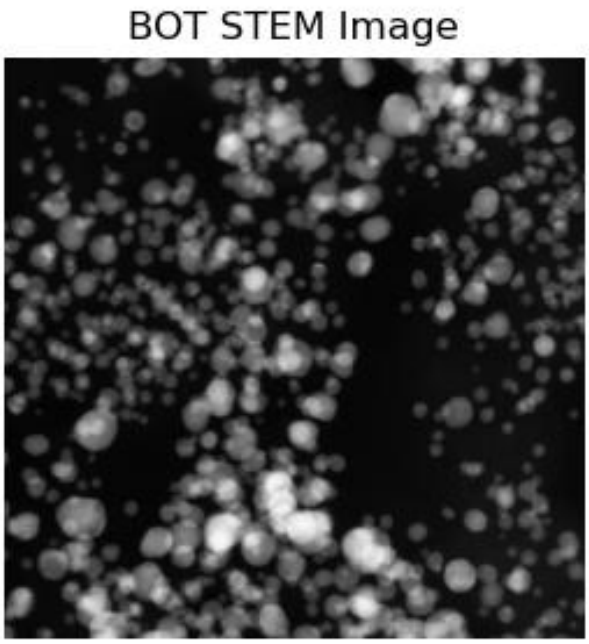

Figure 6.4. BOT image sample from Yu et al.

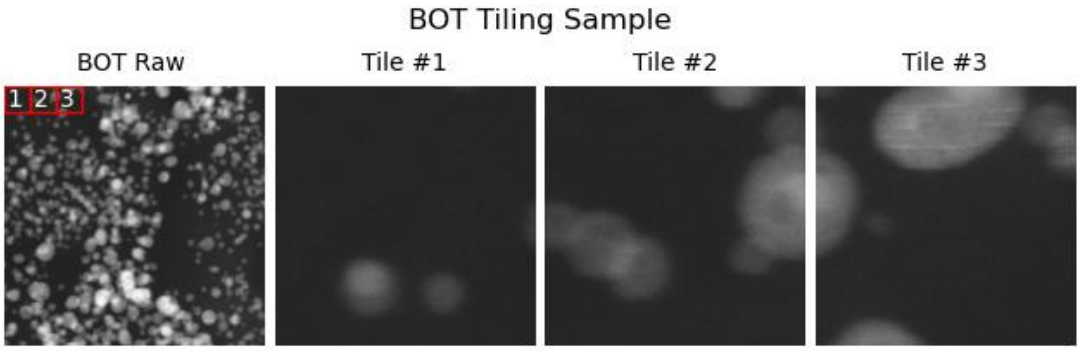

Figure 6.5. Tiling example of the BOT data with the first three tiles shown.

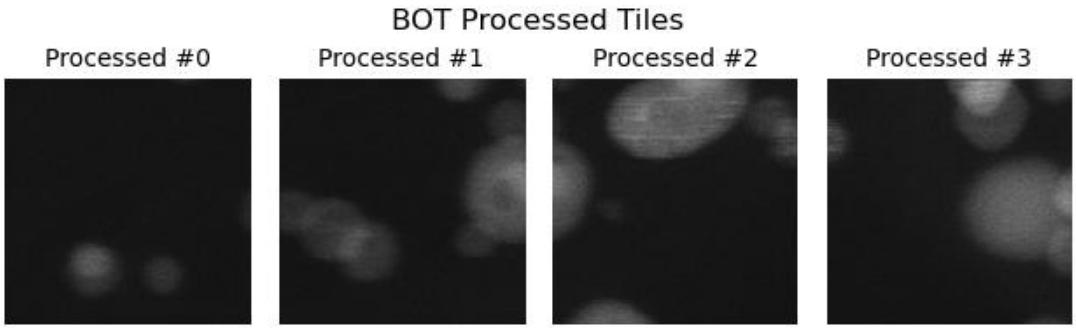

Figure 6.6. Examples of processed BOT images with Laplacian sharpening and CLAHE.

overlapping conditions, especially when dealing with subtle contrast differences and multiple overlapping regions. To address these challenges, Mask R-CNN, a CNN architecture well-suited for instance segmentation problems where multiple objects of the same class need to be distinguished individually was utilized.[244] Mask R-CNN extends the original R-CNN model by combining object detection and pixel-wise segmentation into a unified framework. It proceeds in two stages: first, region proposals are generated to identify candidate areas likely to contain particles; second, a CNN extracts features from each proposed region to refine the bounding boxes, classify objects, and predict a binary segmentation mask for every detected instance. Mask R-CNN's two-stage approach allows it to accurately delineate object boundaries, even when particles touch or partially overlap, a common scenario in the HAADF-STEM simulations.

Each particle in the generated HAADF-STEM dataset is represented by its own binary mask channel, ensuring that overlapping or adjacent particles remain separable. The HAADF-STEM images and masks are given a series of transforms, including flips and stretches. The Mask R-CNN model leverages these multi-channel masks during training, learning to identify individual nanoparticles regardless of their spatial proximity or orientation. After inference, the predicted masks are matched to the ground-truth particle channels using an optimized permutation scheme, minimizing the computational cost of linking each predicted region to the correct particle label. By combining the synthetic labeling approach with the instance segmentation capability of Mask R-CNN, this method enables accurate and automated analysis of complex nanoparticle arrangements. Integrated

the ML pipeline with real-world data bridges the gap between controlled simulation data and real-world microscopy segmentation needs, providing a robust platform for studying particle morphology, aggregation, and spatial distributions in atomically resolved STEM images. This workflow is shown in **Figure 6.7**.

In object detection and particle segmentation, such as Mask R-CNN, several metrics measure how well the model detects particles and how well individual particle boundaries are detected. Precision measures how many of the predicted particles are actual particles expressed as $TP\ /\ (TP\ +\ FP)$ where TP is true positives, and FP is false positives, rewarding models that avoid detecting false objects. Recall captures how many true particles the model found $TP\ /\ (TP\ +\ FN)$, where FN is the false negatives penalizing missed detections. F1 score balances recall and precision, as its harmonic mean $2 \cdot TP\ /\ (2 \cdot TP\ +\ FP + FN)$, giving a single number where high precision can't compensate for poor recall. Finally, Accuracy measures how well the model detects particles overall, expressed as $TP\ /\ (TP\ +\ FP\ +\ FN)$, which equals recall when false positives are zero per level. Values for each of these metrics range from 0 to 1, with 0 having no correct detections and 1 having perfect detections.

Intersection over union (IoU) quantifies mask overlap quality between predicted and ground truth particles, computed as the intersection area (overlapping pixels) divided by the union area (total unique pixels from both masks). Values range 0-1, where 1 means perfect overlap, and 0 means no overlap. In this framework, Mask R-CNN outputs the probability that each pixel is a part of an individual particle. A threshold function set to 0.5

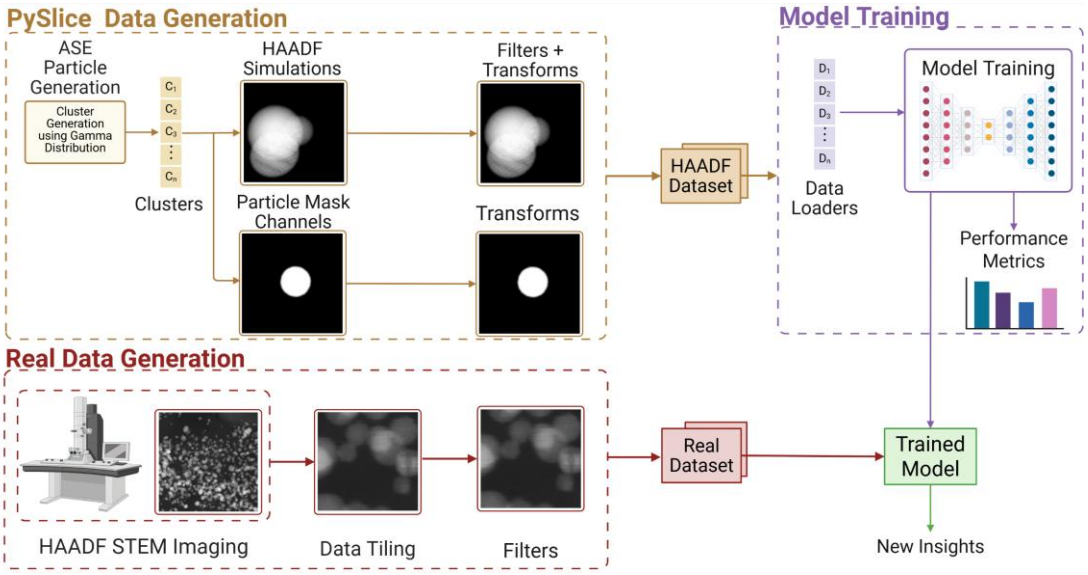

Figure 6.7. ML workflow from PySlice data generation (Orange) to Mask R-CNN training (Purple) to HAADF-STEM imaging (Red) to real data evaluation (Green).

sets all pixels with probabilities above 0.5 as 1 (particle) and below 0.5 as 0 (not particle). The IoU measures how well the particles are detected, then tracks the mean and median IoUs.

Results and Discussion

The Mask R-CNN model, trained exclusively on PySlice-generated multislice HAADF-STEM images, demonstrated strong performance in detecting particles across all 200 test images, as detailed in **Table 6.1** and **Table 6.2** and in **Figure 6.8**. A total of 816 true particles were present in the dataset, while the model predicted 737 particles. Out of these predictions, 732 were correctly identified as true positives, with only five detections classified as false positives. However, the model failed to detect 74 of the particles, resulting in false negatives. These counts translated to a very high precision of 0.993, indicating that nearly all predicted particles corresponded to real particles in the images, with minimal incorrect detections. The recall was 0.897, showing that the model successfully identified the majority of particles but missed a small portion, particularly in challenging configurations with high overlap. The overall F1 score of 0.943 reflects a strong balance between precision and recall, indicating reliable particle detection performance. Additionally, the accuracy was 0.892, meaning that approximately 89% of the total true particles in the dataset were correctly identified by the model. Finally, the mean and median IoU values of 0.9144 for the mean and 0.9309 for the median further confirmed strong spatial agreement between predicted and ground truth masks, typically exceeding 0.85 across the dataset.

Table 6.1. R-CNN testing metrics across all test images.

Predicted Particles	762
True Particles	816
True Positives	732
False Positives	20
False Negatives	74
Precision	0.9738
Recall	0.9093
F1	0.9404
Accuracy	0.8876
Mean IoU	0.9144
Median IoU	0.9309

Table 6.2. R-CNN testing metrics across all test images separated by particle overlap.

Number of Particles	GT	Precision	Recall	F1	Accuracy	Mean IoU	Median IoU
0	73	0.935	1.000	0.998	0.976	0.937	0.940
1	198	0.975	0.955	0.965	0.932	0.936	0.938
2	141	0.974	0.915	0.943	0.893	0.929	0.933
3	148	0.975	0.946	0.960	0.923	0.916	0.928
4	96	0.971	0.833	0.897	0.813	0.894	0.913
5	84	0.970	0.798	0.875	0.779	0.878	0.906
6	42	0.972	0.857	0.911	0.837	0.880	0.903
7	20	0.873	0.900	0.935	0.878	0.817	0.851
8	14	0.967	0.714	0.822	0.697	0.835	0.866

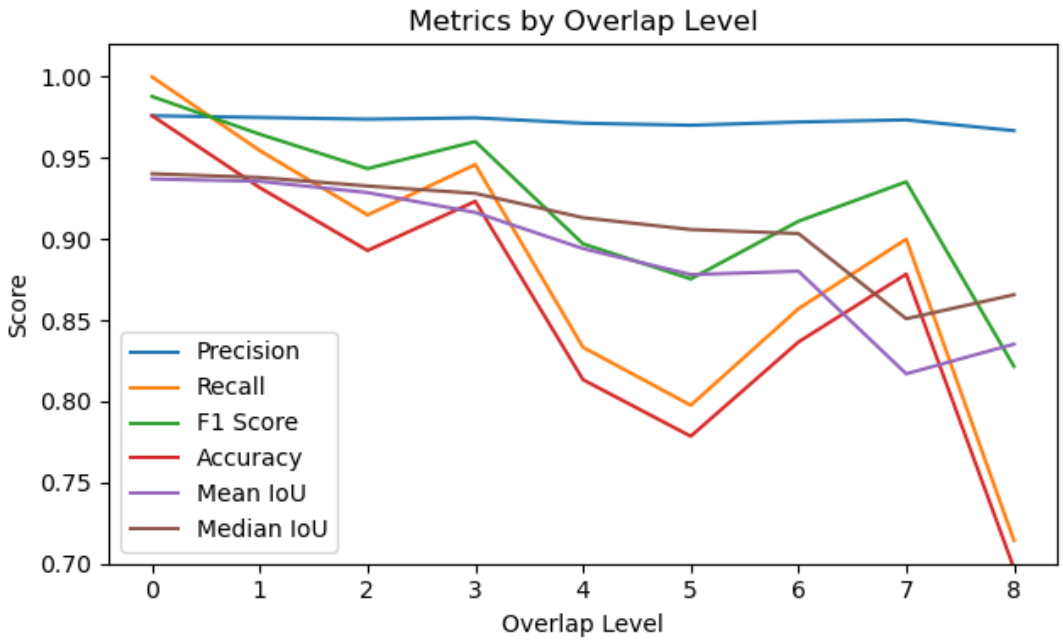

Figure 6.8. Mask R-CNN metrics over all overlap levels. Precision (Blue), recall (Orange), F1-score (Green), accuracy (Red), mean IoU (Purple), and median IoU (Brown).

Performance trends across particle overlap levels, quantified by the number of particles overlapping each individual particle, revealed performance degradation as overlapping increased, as shown in **Table 6.2** and **Figure 6.8**. At overlap level 0 (no overlap), the model performed exceptionally well, achieving a precision of 0.935, near-perfect recall of 0.993, F1 score of 0.963, accuracy of 0.929, and IoU values around 0.94 (mean 0.937, median 0.940). These results indicate that isolated particles were detected very reliably, with few misses and excellent boundary delineation.

For moderate overlaps (levels 1-3, encompassing 198, 141, and 148 of the 737 detected particles respectively), precision remained extremely high, nearly perfect with a precision of 0.974-0.975 reflecting the ability of the model to avoid false positives even as overlapping increased. Recall also remained high, declining from 0.955 at level 1 to 0.946 at level 3. Comparing the balance of precision and recall, the F1 scores of 0.965 to 0.960 and accuracies of 0.932 to 0.923 also remained high. Finally, IoU values were consistently strong (mean 0.916-0.936, median 0.928-0.938), demonstrating that the multichannel masking strategy effectively separated touching particles without compromising edge accuracy.

As overlap intensified (levels 4-6: 96, 84, and 42 particles respectively), recall decreased more noticeably to from 0.798-0.857, with precision remaining more stable at 0.970-0.972 and F1 scores around 0.897-0.911. Accuracies followed suit at 0.813-0.837, while the IoUs dipped slightly but remained viable (mean 0.879-0.894, median 0.903-0.913). The decrease in metrics as overlap increases highlights how dense clustering

primarily challenged detection completeness rather than localization precision, as the model prioritized confident instances amid more subtle contrast gradients.

At the most severe overlaps (levels 7-8, 20 and 14 particles), recall fell to 0.900 and 0.714, producing F1 scores of 0.886 and 0.822, accuracies of 0.878 and 0.697, and IoU around 0.817-0.835. Precision dipped only at level 7 (0.873) before recovering, underscoring that false positives were rare even in highly ambiguous regions. Overall, these results show the model maintained extremely high precision across all overlap levels (average >0.95), with recall limitations confined to heavy crowding; IoU stability (>0.83) affirmed reliable segmentation when detections succeeded, attributing misses to inherent simulation challenges like buried particles rather than architectural shortcomings.

Overall, these results demonstrate that the Mask R-CNN model performs very reliably in identifying Pt nanoparticles, with minimal false detections and a high rate of correct identification across diverse overlap conditions. The consistent high precision (>0.95) and stable IoU (>0.83) across levels 0-8 indicate robust boundary delineation and separation of overlapping instances, even as recall declines in dense regions due to subtle contrast challenges rather than false positives. The model performance validates the simulation-driven training pipeline's ability to produce quantitative morphological analysis suitable for materials characterization workflows.

Table 6.3, **Figure 6.9**, and **Figure 6.10** compare the distributions of particle properties between the ground truth dataset and the particles detected by the Mask R-CNN

Table 6.3. Distribution of particles per image, Feret diameter, area, and aspect ratio of true and detected particles.

Metric	Ground Truth	Predictions
Particles/Image	4.1 ± 2.2	3.7 ± 1.9
Feret Diameter (Å)	36.5 ± 8.4 Å	36.4 ± 8.3 Å
Area (Å²)	1296.4 ± 641.1 Å²	1291.8 ± 642.2 Å²
Aspect Ratio	1.1 ± 0.1	1.1 ± 0.1

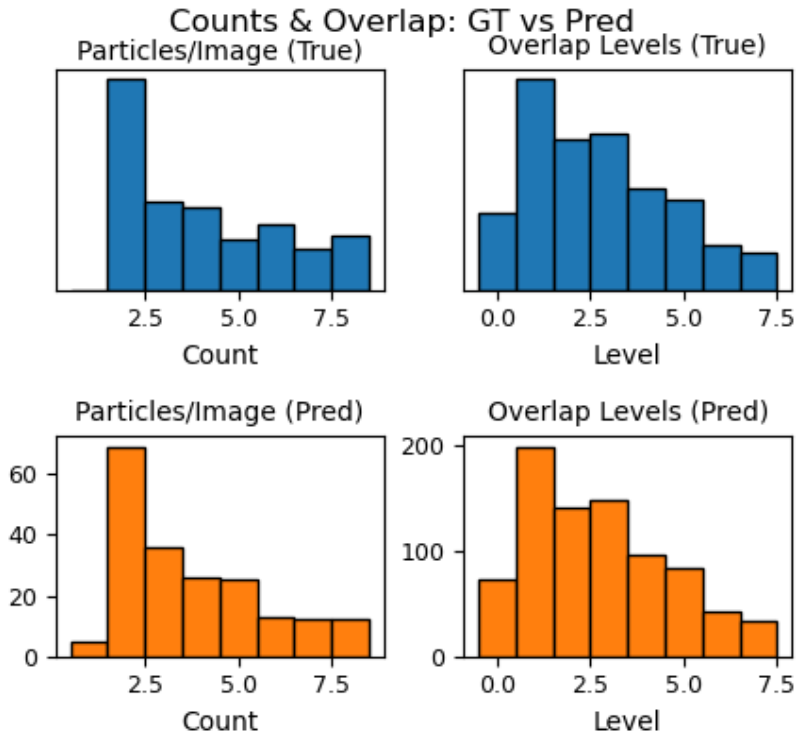

Figure 6.9. True (Blue) versus predicted (Orange) distributions for particles per image and particle overlap.

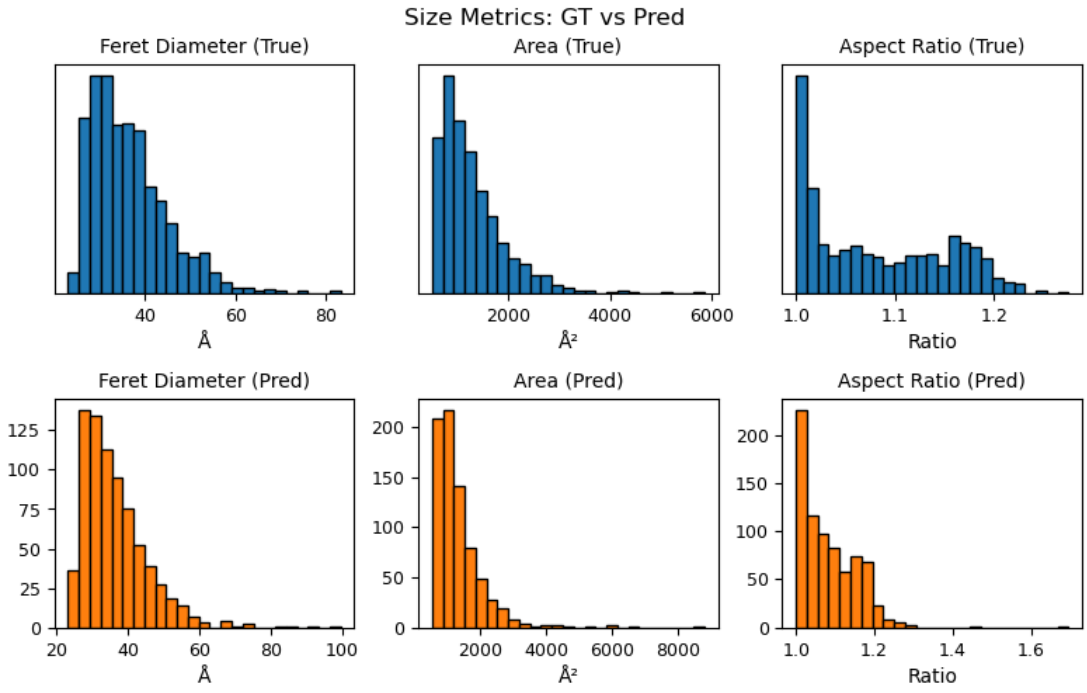

Figure 6.10. True (Blue) versus predicted (Orange) distributions for particle Feret diameter, area, and aspect ratio.

model. The results show that the predicted particle characteristics closely match the ground truth. measurements across all evaluated metrics, indicating that the model preserves the underlying physical characteristics of the nanoparticle population without introducing systematic biases.

The average number of particles per 200×200Å image in the ground truth dataset is 4.1 ± 2.2Å, while the model predicts an average of 3.7 ± 1.9Å particles per image. The slight reduction in predicted particle count is consistent with the overall detection metrics (**Table 6.1** and **Table 6.2**), which showed particularly in highly overlapped regions, leading to a modest underestimation of particle density. The reduction is slightly offset by the few false positives; the model was observed predicting space between particles as a true particle, mainly when the space resembled a particle in shape. However, the overall distribution of particle counts per image remains similar between the ground truth and predicted datasets, as visualized in **Figure 6.9**, with both exhibiting comparable histograms that capture the variability from sparse to moderately dense images. The Feret diameter distributions of the ground truth and predicted particles are nearly identical. Ground truth particles have an average Feret diameter of 36.5 ± 8.4Å, while predicted particles average 36.4 ± 8.3Å shown in **Figure 6.10**. The close agreement between the predicted and ground truth demonstrates that the model accurately identifies particle boundaries and preserves size characteristics, with histograms showing matching right-skewed distributions that include both smaller and larger nanoparticles consistent with the gamma-distributed generation parameters.

Similarly, particle area measurements exhibit excellent agreement, with ground truth at 1296 ± 641 Å² and predictions at 1291 ± 642 Å². The virtually identical means and standard deviations, along with the aligned histogram shapes in **Figure 6.10**, confirm that the model reproduces the full range of particle sizes without preferential detection of small or large particles, enabling reliable quantification of sintering or ripening trends when applied longitudinally.

Aspect ratio distributions are also highly consistent, averaging 1.1 ± 0.1, indicating the model's fidelity in capturing elongated morphologies from scaled elliptical particle generation. **Figure 6.10**'s histograms reveal overlapping profiles, underscoring preserved shape variability essential for distinguishing flat-faceted Pt nanoparticles on carbon supports.

Figure 6.11 presents four samples from the full 200 PySlice test images, showcasing the particle-detection capabilities of Mask R-CNN and the precise pixel-wise segmentation of Pt nanoparticles, with overlaid predicted masks on the raw HAADF-STEM images. These visualizations support the quantitative metrics reported in **Table 6.1** and **Table 6.2**, where the model identified 737 out of 816 ground truth particles with exceptional precision (0.993), solid recall (0.897), F1 score (0.943), accuracy (0.892), and mean/median IoU values consistently above 0.85 across the dataset.

The predicted masks successfully delineate individual particle boundaries even when overlapped, accurately capturing the contrast characteristic of multislice-simulated

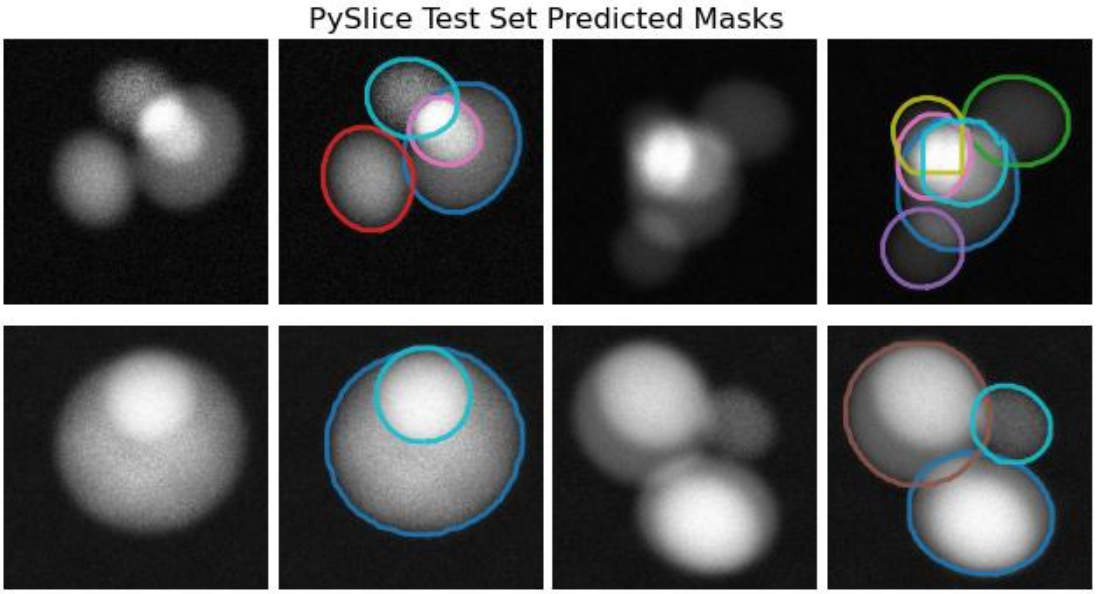

Figure 6.11. Sample of particle detection with the PySlice test data.

Pt nanoparticles, while the false positive detection is in a high overlap region representing the exception to the otherwise near-perfect precision that minimizes spurious detections. The predicted masks visually correlate with the morphological distributions in **Table 6.3**, **Figures 6.9** and **6.10**, where predicted Feret diameters (36.4 ± 8.3Å), areas (1292 ± 642Å²), and aspect ratios (1.1 ± 0.1) preserve ground truth statistics without size or shape bias.

The consistent boundary alignment across overlap configurations validates the multichannel instance masking strategy derived from ASE atomic-hull outlines, with only subtle edge irregularities attributable to Laplacian sharpening and CLAHE preprocessing that do not compromise the quantitative fidelity required for downstream sintering and aggregation analysis. These examples thus bridge the qualitative visualization with the comprehensive detection metrics, establishing the simulation-trained pipeline's reliability for production HAADF-STEM workflows before real PEMFC electrode validation.

Overall, the Mask R-CNN model not only achieves reliable detection metrics (**Table 6.1** and **Table 6.2**) but also preserves the full statistical fidelity of nanoparticle morphology (**Table 6.3**, **Figures 6.9** and **Figure 6.10**), as evidenced by the precise segmentations in **Figure 6.11**. High precision (0.993), stable IoU (>0.85), and unbiased size/shape recovery across a majority of overlap levels demonstrate robust instance separation via simulation-derived multichannel masks, with graceful recall degradation confined to extreme crowding rather than boundary inaccuracies. These simulated data results establish the pipeline's quantitative reliability, setting the stage for validation on real

PEMFC electrode HAADF-STEM images from Yu et al., where transferability to experimental degradation signatures can be directly.

To assess simulation-to-real transferability, morphological metrics from the PySlice-trained Mask R-CNN were extracted from real PEMFC electrode HAADF-STEM images from Yu et al., summarized in **Table 6.4** and in **Figures 6.12** and **6.13**. After tiling $200 \times 200nm$ micrographs (0.2nm/pixel) into non-overlapping 20×20nm patches matching simulation scale, preprocessing with Laplacian sharpening and CLAHE, and excluding edge-detected particles, leaving 45.3% of predicted particles for morphology evaluation. **Table 6.4** reports 4.94 ± 3.2 predicted particles per tile (200×200Å), 33.2 ± 10.8Å Feret diameter of 33.2 ± 10.8Å, 1104 ± 810Å² area, and 1.1 ± 0.1 aspect ratio.

Figure 6.12: histograms illustrate these predicted distributions: particle counts per tile show right-skewed density trends, mirroring the analysis by Yu et al. Feret diameters peak ~25-40Å.[51] Aspect ratios cluster near 1.1, demonstrating elliptical morphologies identical to the PySlice-generated data preserving. Compared to the simulated data in **Table 6.3**, the real predictions shift toward slightly smaller but more variable sizes (33.2 ± 10.8 vs. 36.4 ± 8.3 Å), validating the training pipeline's inclusion of Poisson noise, CLAHE, and blurring that bridges synthetic contrast to experimental subtleties such as detector geometry.

Figure 6.13 showcases predicted edge masks from the PySlice-trained Mask R-CNN applied to representative BOT dataset tiles after tiling, Laplacian sharpening, and

Table 6.4. Distribution of particles per BOT image, Feret diameter, area, and aspect ratio of detected particles.

Metric	Predictions
Non-Edge Particles	45.3%
Particles/Image	4.94 ± 3.2
Feret Diameter (Å)	33.2 ± 10.8
Area (Å²)	1104.5 ± 809.7
Aspect Ratio	1.1 ± 0.1

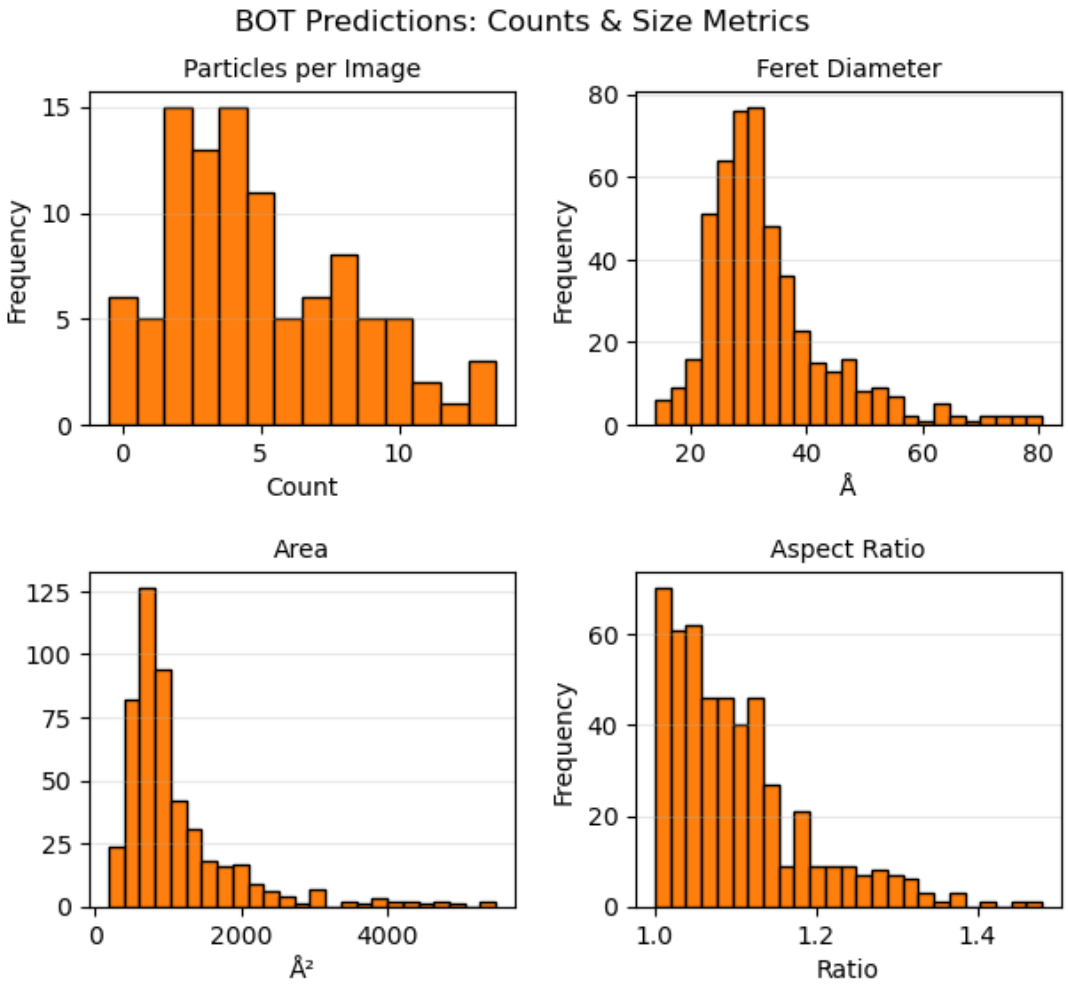

Figure 6.12. Predicted distributions for BOT particle count per images, Feret diameter, area, and aspect ratio.

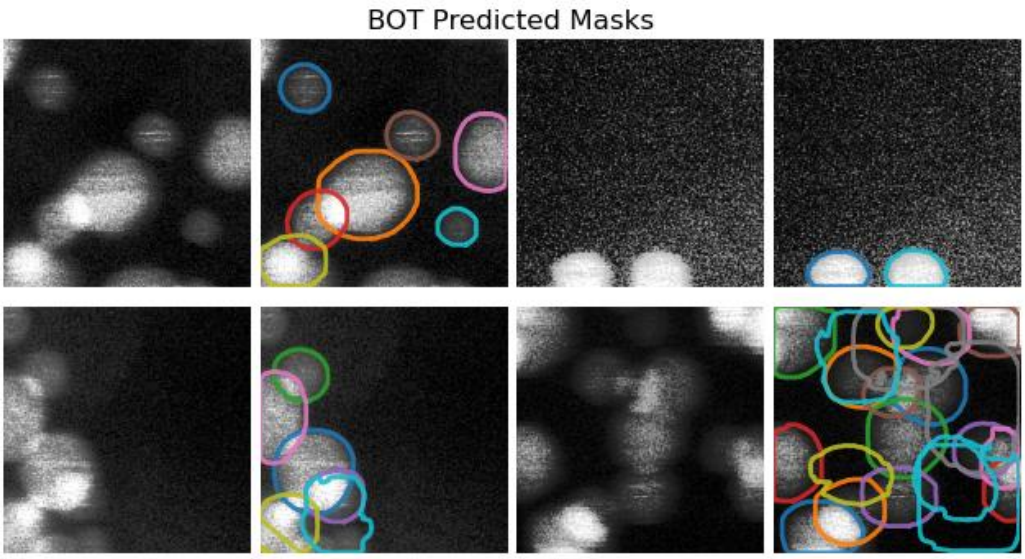

Figure 6.13. Sample of particle detection with the BOT data.

CLAHE preprocessing. These visualizations reveal more jagged boundary contours compared to the smoother PySlice test predictions (**Figure 6.11**), with increased edge artifacts and occasional hallucinations such as minor spurious protrusions or filled gaps in low-contrast regions stemming from real-world experimental challenges like defocus drift, beam-induced noise, and inconsistent detector geometry absent in controlled simulations.

Despite these imperfections, overall mask prediction quality remains strong, successfully delineating overlapping Pt nanoparticles that capture contrast clusters and preserve core morphological trends. The higher hallucination rate represents a small fraction of the overall predictions and is observed in more complex HAADF-STEM images. However, the presented approach still outperforms existing methods in detecting overlapped nanoparticles. These masks demonstrate superior instance separation compared to classical thresholding or watershed methods, which fail on touching particles, yet indicate that the model can still benefit from refinement to improve precision and reduce hallucinations.

The PySlice-trained Mask R-CNN model successfully bridges simulation and real HAADF-STEM imaging, delivering robust nanoparticle detection across 200 simulated test images and transferable performance on PEMFC electrodes with preserved morphological trends despite jagged real-data masks. By generating unbiased training data from ASE-generated clustered and instrument-matched filters, this framework eliminates manual labeling bottlenecks plaguing thresholding, watershed, and prior ML methods, while handling overlaps via multichannel mask instances that classical approaches cannot

resolve. The minor hallucinations and edge artifacts in the real-data predictions highlight the need for refinement, yet they already surpass ex situ manual analysis.

Conclusion

The Mask R-CNN model trained using PySlice-generated data achieved robust detection performance across 200 test images, identifying 737 of 816 ground truth Pt nanoparticles with 0.993 precision, 0.897 recall, and 0.943 F1 score. Precision remained high across overlap levels up to 8, though recall declined from 0.993 (zero overlap) to 0.770 (high crowding), with mean IoU consistently above 0.86. Morphological distributions closely matched ground truth, showing near-identical Feret diameters (36.4 vs 36.5 Å), areas (1292 vs 1296 Å^2), and aspect ratios (1.1 ± 0.1).

The simulation-to-real transfer, validated on PEMFC electrode images, enables automated nanoparticle detection and morphology determination with 4.94 ± 3.2 particles per tile and 33.2 ± 10.8Å (Feret diameter) without manual labeling, directly addressing detection bottlenecks that limit high-throughput HAADF-STEM analysis. By replacing exhaustive real-data annotation with 2,000 physics-accurate PySlice images incorporating probe coherence, Poisson CLAHE, and aberrations, the framework delivers quantitative sizing and overlap-resolved metrology across simulated and experimental datasets. These capabilities establish scalable, simulation-driven workflows for precise characterization of nanoparticle morphology in energy materials, eliminating barriers to ground-truth acquisition while maintaining the statistical fidelity essential for catalyst research.

CHAPTER 7:

CONCLUSION

The central challenge across these energy applications is not simply prediction, but doing so quickly and reliably enough to guide electrolyte formulation and electrode characterization before costly experiments. Across these tasks, the models compare favorably with prior QSPR, computational, and ML baselines while also showing where simpler descriptor-based models remain competitive. The ML models predict EACN, temperature-dependent CMC for different surfactant types, MP using CM-based representations, and non-aqueous solubility using graph-based VAE latent encodings. For MP and non-aqueous solubility, the proposed models reduce errors by 29.9% and 51.2%, respectively, compared to previous QSPR and ML approaches. In practical terms, this enables high-throughput screening: EACN predictions feed directly into HLD-based phase behavior design, CMC predictions guide low-concentration surfactant selection, and operating temperature windows, melting point, and solubility models identify undiluted, room-temperature oils suitable for energy-dense microemulsion electrolytes, and automated STEM analysis provides quantitative nanoparticle statistics in porous electrodes.

The first set of pipelines compares three GNNs and XGBoost for EACN prediction, supporting μE phase stability design using HLD theory. Geometry optimization with MMFF94 and GAFF substantially improves GNN performance. Data from non-optimized geometries showed inaccurate bond lengths and angles, resulting in noticeably worse performance in predicting EACN because these geometric details are tied to

hydrophobicity and therefore to the oil term in HLD. Among the models, CGCNN provides the best accuracy, with an MAE of 1.16, RMSE of 1.55, and R^2 of 0.93, while a descriptor-based XGBoost model trails the top GNNs but remains competitive. These gains over non-optimized geometries indicate that physically meaningful 3D structural encodings are important for reliable EACN prediction. At the same time, there is a clear runtime tradeoff: XGBoost can be trained in less than one second, whereas the GNNs require 28.3 to 206.6 seconds. The accuracy-runtime tradeoff matters because EACN measurements are expensive and slow, so rapid screening is useful for narrowing large candidate sets before committing to experiments.

For CMC, the work augments GNNs by including temperature as a surrogate node feature, enabling direct prediction of temperature-dependent micellization across cationic, anionic, and nonionic surfactants. Because temperature is not encoded in a static molecular graph, incorporating it explicitly improves the model's ability to represent temperature-dependent micellization. With this augmented representation, the CGCNN model achieves an MAE of 0.261 logM, an RMSE of 0.428 logM, and an R^2 of 0.835. These results show that graph-based models with explicit temperature information perform well for CMC across surfactant classes and operating temperatures. As with EACN, the GNNs incur longer runtimes than XGBoost and Gaussian processes, which train in tens of milliseconds to a couple of seconds, while the GNNs can take up to 13 minutes. The runtime-accuracy tradeoff suggests a workflow in which fast descriptor models are used for broad screening and GNNs are reserved for narrower candidate sets that warrant higher-fidelity predictions.

The MP study develops a CM-based encoding approach that, when augmented with RDKit, PaDEL, and ECFP descriptors, yields accurate MP predictions using the compiled dataset, which is the largest organic MP dataset compiled to date. The best model achieves an MAE of 18.4°C, an RMSE of 27.1°C, and an R^2 of 0.92. Compared with prior state-of-the-art QSPR and ML models, this corresponds to a 29.9% reduction in MAE. The SHAP analysis shows that the most important features are ECFPs, HBD counts, TPSA, and CM-based descriptors, while purely topological descriptors such as BertzCT contribute less. The descriptor ranking suggests that physically motivated features capturing intermolecular forces and specific interactions are especially informative for melting behavior in this dataset. In practice, this supports filtering redox-active oils by their likelihood of remaining liquid at room temperature without additional solvent, which is important for maximizing energy density in μE-based electrolytes.

The solubility work uses VAE-based graph encoders trained on ZINC, QM9, and their combination to produce learned latent representations of solutes and solvents that serve as alternative descriptors for non-aqueous solubility prediction. These latent vectors, combined with RDKit and PaDEL descriptors and temperature, are fed into an XGBoost model trained on the BigSolDB dataset, the largest solubility dataset to date, which contains solute, solvent, and temperature information. The resulting model achieves an MAE of 0.210 logS, an RMSE of 0.308 logS, and an R^2 of 0.925, representing a 51.2% improvement over previous ML baselines and a 63% improvement over the GSE. These gains show that learned graph encodings capture interaction patterns that complement

hand-crafted descriptors and outperform simpler empirical equations in this dataset. For BEES², this means the model can be used to estimate how much of a candidate redox-active species will dissolve in a given organic or mixed solvent at different temperatures before experiments. That, in turn, enables formulation strategies that minimize oil dilution while still forming homogeneous, stable μE, increasing energy density.

For PCEs, a Mask R-CNN pipeline for nanoparticle detection in STEM images of electrodes was developed. The Mask R-CNN model was trained using PySlice-simulated data that includes realistic noise, particle overlap, and unbiased ground-truth labels. The trained model achieves precision, recall, F1 score, and both mean and median IoU greater than 0.9 on synthetic test data, and the resulting particle size and shape distributions closely align with those measured in real Pt/C catalyst STEM images. The analysis of Feret diameter distributions and other morphological metrics shows good agreement with ground truth, where simple thresholding and watershed segmentation fail, especially in regions where particles overlap or have low contrast. The results suggest that a model trained on simulated data can transfer well to real STEM images. For electrode development, this capability offers a route to transform labor-intensive, low-throughput manual segmentation into automated, high-throughput characterization once new images are acquired.

Taken together, these results link ML-driven prediction of electrolyte properties with automated nanoparticle morphology analysis into a unified framework for high-throughput discovery in energy storage. Across molecular screening, μE formulation, and electrode metrology, these models reduce reliance on costly, slow, and low-throughput

experiments. Instead of measuring every property and analyzing every image by hand, researchers can use the models to triage large design spaces and focus their experimental efforts where they are most likely to pay off. In the specific context of BEES², this means enabling rapid exploration of microemulsion compositions that achieve type IV behavior with high oil loading, selecting redox-active oils and surfactants that remain liquid and conductive under operating conditions, and quantifying catalyst utilization and degradation mechanisms in porous electrodes. By validating these pipelines on real chemical systems and imaging data, the study moves beyond proof of concept toward a practical, interpretable set of tools that help chemists and materials scientists design better electrolytes and electrodes on timescales relevant to iterative materials development.

REFERENCES

(1) U.S. Department of Energy, O. of S. *Breakthrough Electrolytes for Energy Storage and Systems (BEES2) EFRC Technical Summary 2022--11*; 2022.

(2) Savinell, R. *Breakthrough Electrolytes for Energy Storage and Systems (BEES2)*. U.S. Department of Energy, Office of Science, Basic Energy Sciences.

(3) Winsberg, J.; Hagemann, T.; Janoschka, T.; Hager, M. D.; Schubert, U. S. Redox-Flow Batteries: From Metals to Organic Redox-Active Materials. *Angewandte Chemie - International Edition*. 2017. https://doi.org/10.1002/anie.201604925.

(4) Pan, F.; Wang, Q. Redox Species of Redox Flow Batteries: A Review. *Molecules*. 2015. https://doi.org/10.3390/molecules201119711.

(5) Barth, B. A.; Imel, A.; Nelms, K. M.; Goenaga, G. A.; Zawodzinski, T. Microemulsions: Breakthrough Electrolytes for Redox Flow Batteries. *Front. Chem.* **2022**, *10*, 831200.

(6) Shen, X.; Sinclair, N.; Wainright, J.; Imel, A.; Barth, B.; Zawodzinski, T.; Savinell, R. F. A Study of Ferrocene Diffusion in Toluene/Tween 20/1-Butanol/Water Microemulsions for Redox Flow Battery Applications. *J. Electrochem. Soc.* **2021**, *168* (6), 060539. https://doi.org/10.1149/1945-7111/ac0b26.

(7) Emma Hollmann. Factors Influencing Kinetics of Electrode Processes in Vanadium Redox Flow Batteries, University of Tennessee-Knoxville, Knoxville, 2015.

(8) Ahn, S.; Yun, A.; Ko, D.; Singh, V.; Joo, J. M.; Byon, H. R. Organic Redox Flow Batteries in Non-Aqueous Electrolyte Solutions. *Chemical Society Reviews*. Royal Society of Chemistry November 27, 2024, pp 742–789. https://doi.org/10.1039/d4cs00585f.

(9) Mouselly, M.; Alawadhi, H.; Senthilkumar, S. T. Current Status of Ferro-/Ferricyanide for Redox Flow Batteries. *Current Opinion in Electrochemistry*. 2024. https://doi.org/10.1016/j.coelec.2024.101581.

(10) Lim, H. S.; Chae, S.; Yan, L.; Li, G.; Feng, R.; Shin, Y.; Nie, Z.; Sivakumar, B. M.; Zhang, X.; Liang, Y.; Bazak, D. J.; Shutthanandan, V.; Murugesan, V.; Kim, S.; Wang, W. Crosslinked Polyethyleneimine Gel Polymer Interface to Improve Cycling Stability of RFBs. *Energy Material Advances* **2022**, *2022*. https://doi.org/10.34133/2022/9863679.

(11) Ashok, A.; Kumar, A. A Comprehensive Review of Metal-Based Redox Flow Batteries: Progress and Perspectives. *Green Chemistry Letters and Reviews*. 2024. https://doi.org/10.1080/17518253.2024.2302834.

(12) Sharmoukh, W. Redox Flow Batteries as Energy Storage Systems: Materials, Viability, and Industrial Applications. *RSC Advances*. 2025. https://doi.org/10.1039/d5ra00296f.

(13) Madhu, R.; Kusmartsev, F. V.; Sakthivel, M.; Vediyappan, V.; Ahn, H. J.; Ho, K. C.; Tsai, H. S. Multi-Fractal Nanoporous Carbon Sphere-Decorated Graphite Felt Electrodes for Vanadium Redox Flow Batteries. *ACS Appl. Energy Mater.* **2025**, *8* (15). https://doi.org/10.1021/acsaem.5c01341.

(14) Derr, I.; Przyrembel, D.; Schweer, J.; Fetyan, A.; Langner, J.; Melke, J.; Weinelt, M.; Roth, C. Electroless Chemical Aging of Carbon Felt Electrodes for the All-Vanadium Redox Flow Battery (VRFB) Investigated by Electrochemical Impedance and X-Ray Photoelectron Spectroscopy. *Electrochim. Acta* **2017**, *246*. https://doi.org/10.1016/j.electacta.2017.06.050.

(15) Duduta, M. Semi-Solid Redox Flow Battery. *Thesis* **2010**.

(16) Jung, S. Y.; Han, J.; Choi, S. K.; Cho, S. Y.; Won, J. H.; Choi, J.; Yang, M. Uniform Lithium Deposition Using Cu Teepee Structures for Anode-Free Lithium Metal Batteries. *Chemical Engineering Journal* **2025**, *522*. https://doi.org/10.1016/j.cej.2025.167302.

(17) Small, L. J.; Pratt, H. D.; Anderson, T. M. Crossover in Membranes for Aqueous Soluble Organic Redox Flow Batteries. *J. Electrochem. Soc.* **2019**, *166* (12). https://doi.org/10.1149/2.0681912jes.

(18) Ferreira, R. B.; Falcão, D. S.; Pinto, A. M. F. R. Simulation of Membrane Chemical Degradation in a Proton Exchange Membrane Fuel Cell by Computational Fluid

Dynamics. *Int. J. Hydrogen Energy* **2021**, *46* (1). https://doi.org/10.1016/j.ijhydene.2020.09.179.

(19) Volodin, I. A.; Wulf, K.; Tzschoeckell, F.; Stumpf, S.; Hoeppener, S.; Fritz, N.; Morales-Reyes, C. F.; Wichard, T.; Ueberschaar, N.; Stolze, C.; Hager, M. D.; Schubert, U. S. Evaluation of in Situ Thermal Stability Assessment for Flow Batteries and Deeper Investigation of the Ferrocene Co-Polymer. *J. Mater. Chem. A Mater.* **2024**, *12* (8). https://doi.org/10.1039/d3ta05809c.

(20) Tartaro, G.; Mateos, H.; Schirone, D.; Angelico, R.; Palazzo, G. Microemulsion Microstructure(s): A Tutorial Review. *Nanomaterials* **2020**, *10* (9). https://doi.org/10.3390/nano10091657.

(21) Acosta, E. The HLD–NAC Equation of State for Microemulsions Formulated with Nonionic Alcohol Ethoxylate and Alkylphenol Ethoxylate Surfactants. *Colloids Surf. A Physicochem. Eng. Asp.* **2008**, *320*, 193–204. https://doi.org/10.1016/j.colsurfa.2008.01.049.

(22) Furth, N. R.; Imel, A. E.; Zawodzinski, T. A. Comparison of Machine Learning Approaches for Prediction of the Equivalent Alkane Carbon Number for Microemulsions Based on Molecular Properties. *J. Phys. Chem. A* **2024**, *128* (32), 6763–6773. https://doi.org/10.1021/acs.jpca.4c00936.

(23) Do, L. D.; Attaphong, C.; Scamehorn, J. F.; Sabatini, D. A. Detergency of Vegetable Oils and Semi-Solid Fats Using Microemulsion Mixtures of Anionic Extended Surfactants: The HLD Concept and Cold Water Applications. *J. Surfactants Deterg.* **2015**, *18*, 373–382.

(24) Acosta, E.; Szekeres, E.; Sabatini, D. A.; Harwell, J. H. Net-Average Curvature Model for Solubilization and Supersolubilization in Surfactant Microemulsions. *Langmuir* **2003**, *19* (1), 186–195. https://doi.org/10.1021/la026168a.

(25) Spernath, A.; Yaghmur, A.; Aserin, A.; Hoffman, R. E.; Garti, N. Food-Grade Microemulsions Based on Nonionic Emulsifiers: Media To Enhance Lycopene Solubilization. *J. Agric. Food Chem.* **2002**, *50* (23), 6917–6922. https://doi.org/10.1021/jf025762n.

(26) Budhathoki, M.; Hsu, T.-P.; Lohateeraparp, P.; Roberts, B. L.; Shiau, B.-J.; Harwell, J. H. Design of an Optimal Middle Phase Microemulsion for Ultra High Saline Brine Using Hydrophilic Lipophilic Deviation (HLD) Method. *Colloids Surf. A Physicochem. Eng. Asp.* **2016**, *488*, 36–45. https://doi.org/https://doi.org/10.1016/j.colsurfa.2015.09.066.

(27) Imel, A. E.; Barth, B.; Hayes, D. G.; Dadmun, M.; Zawodzinski, T. Microemulsions as Emerging Electrolytes: The Correlation of Structure to Electrochemical Response. *ACS Appl. Mater. Interfaces* **2022**, *14* (17), 20179–20189.

(28) Liu, Y.; Chen, S.; Sheng, L.; Wang, M.; Liu, H. The Phase Behavior and Solubilization Ability of Nonionic Surfactant-Distillate Fraction of Crude Oil Microemulsion System. *Colloids Surf. A Physicochem. Eng. Asp.* **2020**, *603*. https://doi.org/10.1016/j.colsurfa.2020.125181.

(29) Jin, L.; Budhathoki, M.; Jamili, A.; Li, Z.; Luo, H.; Ben Shiau, B. J.; Delshad, M.; Harwell, J. H. Predicting Microemulsion Phase Behavior Using Physics Based HLD-NAC Equation of State for Surfactant Flooding. *J. Pet. Sci. Eng.* **2017**, *151*, 213–223. https://doi.org/https://doi.org/10.1016/j.petrol.2016.12.035.

(30) Sharma, V. K.; Hayes, D. G.; Gupta, S.; Urban, V. S.; O'Neill, H. M.; Pingali, S. V.; Ohl, M.; Mamontov, E. Incorporation of Melittin Enhances Interfacial Fluidity of Bicontinuous Microemulsions. *Journal of Physical Chemistry C* **2019**, *123* (17). https://doi.org/10.1021/acs.jpcc.9b00103.

(31) Acosta, E.; Kiran, S.; Hammond, C. The HLD-NAC Model for Extended Surfactant Microemulsions. *J. Surfactants Deterg.* **2012**, *15*. https://doi.org/10.1007/s11743-012-1343-2.

(32) Peng, J.; Xiao, Y.; Imel, A.; Barth, B. A.; Cantillo, N. M.; Nelms, K. M. K.; Zawodzinski, T. A. Electrolyte Effects on the Electrochemical Performance of Microemulsions. *Electrochim. Acta* **2021**, *393*. https://doi.org/10.1016/j.electacta.2021.139048.

(33) Peng, J.; Cantillo, N. M.; Xiao, Y.; Nelms, K. M.; Roberts, L. S.; Goenaga, G.; Imel, A.; Barth, B. A.; Dadmun, M.; Hayes, D. G.; Zawodzinski, T. Decoupling Conductivity and Solubility in Electrolytes Using Microemulsions. *J. Electrochem. Soc.* **2021**, *168* (8), 80502.

(34) Nazar, M. F.; Shah, S. S.; Khosa, M. A. Microemulsions in Enhanced Oil Recovery: A Review. *Pet. Sci. Technol.* **2011**, *29*, 1353–1365.

(35) De Gennes, P. G.; Taupin, C. Microemulsions and the Flexibility of Oil/Water Interfaces. *J. Phys. Chem.* **1982**, *86* (13), 2294–2304. https://doi.org/10.1021/j100210a011.

(36) Fan, L.; Wang, Z.; Lu, Y.; Zhou, J. An Adversarial Learning Approach for Super-Resolution Enhancement Based on AgCl@Ag Nanoparticles in Scanning Electron Microscopy Images. *Nanomaterials* **2021**, *11* (12). https://doi.org/10.3390/nano11123305.

(37) Tchakalova, V.; Bailly, C.; Fieber, W. Food-Grade Bicontinuous Microemulsions. *Flavour Fragr. J.* **2014**, *29* (1), 67–74. https://doi.org/https://doi.org/10.1002/ffj.3181.

(38) Eastoe, J.; Warne, B. Nanoparticle and Polymer Synthesis in Microemulsions. *Curr. Opin. Colloid Interface Sci.* **1996**, *1* (6), 800–805. https://doi.org/10.1016/S1359-0294(96)80084-7.

(39) Boonme, P. Applications of Microemulsions in Cosmetics. *J. Cosmet. Dermatol.* **2007**, *6* (4), 223–228. https://doi.org/https://doi.org/10.1111/j.1473-2165.2007.00337.x.

(40) Solomou, A.; Zhao, G.; Boluki, S.; Joy, J. K.; Qian, X.; Karaman, I.; Arróyave, R.; Lagoudas, D. C. Multi-Objective Bayesian Materials Discovery: Application on the Discovery of Precipitation Strengthened NiTi Shape Memory Alloys through Micromechanical Modeling. *Mater. Des.* **2018**, *160*. https://doi.org/10.1016/j.matdes.2018.10.014.

(41) Schwuger, M.-J.; Stickdorn, K.; Schomaecker, R. Microemulsions in Technical Processes. *Chem. Rev.* **1995**, *95* (4), 849–864. https://doi.org/10.1021/cr00036a003.

(42) Khorsandi, S.; Johns, R. T. Robust Flash Calculation Algorithm for Microemulsion Phase Behavior. *J. Surfactants Deterg.* **2016**, *19* (6), 1273–1287. https://doi.org/https://doi.org/10.1007/s11743-016-1877-9.

(43) Baccile, N.; Poirier, A. Microbial Bio-Based Amphiphiles (Biosurfactants): General Aspects on Critical Micelle Concentration, Surface Tension, and Phase Behavior. In *Biosurfactants: Research and Development*; 2023. https://doi.org/10.1016/B978-0-323-91697-4.00001-6.

(44) Acosta, E. Engineering Cosmetics Using the Net-Average-Curvature (NAC) Model. *Current Opinion in Colloid and Interface Science*. 2020. https://doi.org/10.1016/j.cocis.2020.05.005.

(45) Hoar, T. P.; Schulman, J. H. Transparent Water-in-Oil Dispersions: The Oleopathic Hydro-Micelle. *Nature* **1943**, *152* (3847), 102–103. https://doi.org/10.1038/152102a0.

(46) Winsor, P. A. Hydrotropy, Solubilisation and Related Emulsification Processes. *Transactions of the Faraday Society* **1948**, *44* (0), 376–398. https://doi.org/10.1039/TF9484400376.

(47) Acosta, E.; Bhakta, A. The HLD-NAC Model for Mixtures of Ionic and Nonionic Surfactants. *J. Surfactants Deterg.* **2009**, *12*, 7–19. https://doi.org/10.1007/s11743-008-1092-4.

(48) Jin, L.; Jamili, A.; Li, Z.; Lu, J.; Luo, H.; Ben Shiau, B. J.; Delshad, M.; Harwell, J. H. Physics Based HLD–NAC Phase Behavior Model for Surfactant/Crude Oil/Brine Systems. *J. Pet. Sci. Eng.* **2015**, *136*, 68–77. https://doi.org/https://doi.org/10.1016/j.petrol.2015.10.039.

(49) Kiran Sumit K. and Nace, V. M. and S. M. A. and M. K. A. and M. J. and S. L. and A. E. J. The HLD Study of Surfactant Partitioning for Oilfield Corrosion Inhibitors.

J. Surfactants Deterg. **2014**, *17* (6), 1193–1201. https://doi.org/10.1007/s11743-014-1631-0.

(50) Tang, W.; Maimaiti, P.; Shao, H.; Que, T.; Bai, S. Optimizing Middle-Phase Microemulsions for Enhanced Oil Recovery: An HLD-NAC Theoretical Framework and Experimental Validation. *ACS Omega* **2025**, *10* (44). https://doi.org/10.1021/acsomega.5c06318.

(51) Yu, H.; Zachman, M. J.; Reeves, K. S.; Park, J. H.; Kariuki, N. N.; Hu, L.; Mukundan, R.; Neyerlin, K. C.; Myers, D. J.; Cullen, D. A. Tracking Nanoparticle Degradation across Fuel Cell Electrodes by Automated Analytical Electron Microscopy. *ACS Nano* **2022**, *16* (8). https://doi.org/10.1021/acsnano.2c02307.

(52) Delforce, L.; Duprat, F.; Ploix, J.-L.; Ontiveros, J. F.; Goussard, V.; Nardello-Rataj, V.; Aubry, J.-M. Fast Prediction of the Equivalent Alkane Carbon Number Using Graph Machines and Neural Networks. *ACS Omega* **2022**, *7* (43), 38869–38881. https://doi.org/10.1021/acsomega.2c04592.

(53) Jang, S. H.; Pope, G. A. Microemulsion Phase Behavior of Live Crude Oil and Revisiting the EACN Framework for Crude Oils. *Colloids Surf. A Physicochem. Eng. Asp.* **2023**, *670*. https://doi.org/10.1016/j.colsurfa.2023.131565.

(54) Noh, T. H.; Kim, M. Y.; Kim, D. H.; Yang, S. H.; Lee, J. H.; Park, H. S.; Noh, H. S.; Lee, M. S.; Kim, H. S. Electrochemical Studies of Carbon Felt Electrode

Modified under Airless Conditions for Redox Flow Batteries. *Journal of Electrochemical Science and Technology* **2017**, *8* (2). https://doi.org/10.5229/JECST.2017.8.2.155.

(55) Bergström, L. M. A Theoretical Investigation of the Influence of the Second Critical Micelle Concentration on the Solubilization Capacity of Surfactant Micelles. *AIP Adv.* **2018**, *8* (5). https://doi.org/10.1063/1.5027062.

(56) Barth, B. A.; Wood, A. L.; Albenice, P. J.; Imel, A. E.; Sharma, B.; Zawodzinski, T. A. Surfactant Effects in Porous Electrodes for Microemulsion Redox Flow Batteries. *J. Electrochem. Soc.* **2024**, *171* (5). https://doi.org/10.1149/1945-7111/ad4ac5.

(57) Hosseini, S.; Lao-atiman, W.; Han, S. J.; Arpornwichanop, A.; Yonezawa, T.; Kheawhom, S. Discharge Performance of Zinc-Air Flow Batteries Under the Effects of Sodium Dodecyl Sulfate and Pluronic F-127. *Sci. Rep.* **2018**, *8* (1). https://doi.org/10.1038/s41598-018-32806-3.

(58) Perinelli, D. R.; Cespi, M.; Lorusso, N.; Palmieri, G. F.; Bonacucina, G.; Blasi, P. Surfactant Self-Assembling and Critical Micelle Concentration: One Approach Fits All? *Langmuir* **2020**, *36* (21). https://doi.org/10.1021/acs.langmuir.0c00420.

(59) Cheng, Y.; Hall, D. M.; Boualavong, J.; Hickey, R. J.; Lvov, S. N.; Gorski, C. A. Influence of Hydrotropes on the Solubilities and Diffusivities of Redox-Active

Organic Compounds for Aqueous Flow Batteries. *ACS Omega* **2021**, *6* (45). https://doi.org/10.1021/acsomega.1c05133.

(60) Li, L.; Qu, J.; Liu, W.; Peng, B.; Cong, S.; Yu, H.; Zhang, B.; Li, Y. Advancements in Characterization Techniques for Microemulsions: From Molecular Insights to Macroscopic Phenomena. *Molecules*. 2024. https://doi.org/10.3390/molecules29122901.

(61) Shahu, S. G.; Wadetwar, R. N.; Dixit, G. R. Development of Microemulsion for Solubility Enhancement of Poorly Water Soluble Drug Valsartan. *Int. J. Pharm. Sci. Rev. Res.* **2013**, *22* (2).

(62) Leng, P.; Zhang, Z.; Li, Q.; Zhao, M.; Pan, G. Microemulsion Formulation of Carbendazim and Its in Vitro Antifungal Activities Evaluation. *PLoS One* **2014**, *9* (10). https://doi.org/10.1371/journal.pone.0109580.

(63) Klier, J.; Tucker, C. J.; Kalantar, T. H.; Green, D. P. Properties and Applications of Microemulsions. *Advanced Materials* **2000**, *12* (23). https://doi.org/10.1002/1521-4095(200012)12:23<1751::AID-ADMA1751>3.0.CO;2-I.

(64) Mustafa, I.; Susantyoko, R.; Wu, C. H.; Ahmed, F.; Hashaikeh, R.; Almarzooqi, F.; Almheiri, S. Nanoscopic and Macro-Porous Carbon Nano-Foam Electrodes with Improved Mass Transport for Vanadium Redox Flow Batteries. *Sci. Rep.* **2019**, *9* (1). https://doi.org/10.1038/s41598-019-53491-w.

(65) Köble, K.; Schilling, M.; Eifert, L.; Bevilacqua, N.; Fahy, K. F.; Atanassov, P.; Bazylak, A.; Zeis, R. Revealing the Multifaceted Impacts of Electrode Modifications for Vanadium Redox Flow Battery Electrodes. *ACS Appl. Mater. Interfaces* **2023**, *15* (40). https://doi.org/10.1021/acsami.3c07940.

(66) Ding, C.; Shen, Z.; Zhu, Y.; Cheng, Y. Insights into the Modification of Carbonous Felt as an Electrode for Vanadium Redox Flow Batteries. *Materials*. 2023. https://doi.org/10.3390/ma16103811.

(67) Yamamoto, H. Estimation of EACN (Equivalent Alkane Carbon Number) with HSP. *Pirika*. https://pirika.com/NewHP/PirikaE3/EACN.html.

(68) Acree, W. E.; Chong, W.-K.; Lang, A. S. I. D.; Mozafari, H. Estimating Equivalent Alkane Carbon Number Using Abraham Solute Parameters. *Liquids* **2022**, *2* (4). https://doi.org/10.3390/liquids2040019.

(69) Bouton, F.; Durand, M.; Nardello-Rataj, V.; Serry, M.; Aubry, J.-M. Classification of Terpene Oils Using the Fish Diagrams and the Equivalent Alkane Carbon (EACN) Scale. *Colloids Surf. A Physicochem. Eng. Asp.* **2009**, *338* (1), 142–147. https://doi.org/https://doi.org/10.1016/j.colsurfa.2008.05.027.

(70) Lukowicz, T.; Illous, E.; Nardello-Rataj, V.; Aubry, J. M. Prediction of the Equivalent Alkane Carbon Number (EACN) of Aprotic Polar Oils with COSMO-

RS Sigma-Moments. *Colloids Surf. A Physicochem. Eng. Asp.* **2018**, *536*, 53–59. https://doi.org/10.1016/J.COLSURFA.2017.07.068.

(71) Lukowicz, T.; Benazzouz, A.; Nardello-Rataj, V.; Aubry, J.-M. Rationalization and Prediction of the Equivalent Alkane Carbon Number (EACN) of Polar Hydrocarbon Oils with COSMO-RS σ-Moments. *Langmuir* **2015**, *31* (41), 11220–11226. https://doi.org/10.1021/acs.langmuir.5b02545.

(72) Ontiveros, J. F.; Pierlot, C.; Catté, M.; Molinier, V.; Pizzino, A.; Salager, J.-L.; Aubry, J.-M. Classification of Ester Oils According to Their Equivalent Alkane Carbon Number (EACN) and Asymmetry of Fish Diagrams of C10E4/Ester Oil/Water Systems. *J. Colloid Interface Sci.* **2013**, *403*, 67–76. https://doi.org/https://doi.org/10.1016/j.jcis.2013.03.071.

(73) Aubry, J.-M.; Ontiveros, J. F.; Salager, J.-L.; Nardello-Rataj, V. Use of the Normalized Hydrophilic-Lipophilic-Deviation (HLDN) Equation for Determining the Equivalent Alkane Carbon Number (EACN) of Oils and the Preferred Alkane Carbon Number (PACN) of Nonionic Surfactants by the Fish-Tail Method (FTM). *Adv. Colloid Interface Sci.* **2020**, *276*, 102099. https://doi.org/https://doi.org/10.1016/j.cis.2019.102099.

(74) Zarate-Muñoz, S.; de Vasconcelos, F.; Myint-Myat, K.; Minchom, J.; Acosta, E. A Simplified Methodology to Measure the Characteristic Curvature (Cc) of Alkyl

Ethoxylate Nonionic Surfactants. *J. Surfactants Deterg.* **2016**, *19* (2), 249–263. https://doi.org/https://doi.org/10.1007/s11743-016-1787-x.

(75) Sinha, S.; Yang, C.; Wu, E.; Acree, W. E. Abraham Solvation Parameter Model: Examination of Possible Intramolecular Hydrogen-Bonding Using Calculated Solute Descriptors. *Liquids* **2022**, *2* (3). https://doi.org/10.3390/liquids2030009.

(76) Yoga, H. F.; Gasimli, N. R.; Johns, R. T. Reliable Equivalent Alkane Carbon Number Determination for Dead and Live Crudes in Microemulsion Systems. *SPE Journal* **2024**, *29* (9). https://doi.org/10.2118/221470-PA.

(77) Poša, M.; Bjedov, S.; Škorić, D.; Sakač, M. Micellization Parameters (Number Average, Aggregation Number and Critical Micellar Concentration) of Bile Salt 3 and 7 Ethylidene Derivatives: Role of the Steroidal Skeleton II. *Biochim. Biophys. Acta Gen. Subj.* **2015**, *1850* (7). https://doi.org/10.1016/j.bbagen.2015.03.010.

(78) Chang, C. Y.; Wang, S. J.; Liu, I. J.; Chiu, Y. C. A Simple Method for Determining the Critical Micellar Concentration of a Surfactant. *Journal of the Chinese Chemical Society* **1987**, *34* (3). https://doi.org/10.1002/jccs.198700037.

(79) Mukerjee, P.; Mysels, K. J. Critical Micelle Concentrations of Aqueous Surfactant Systems. *J. Pharm. Sci.* **1972**, *61* (2), 319. https://doi.org/10.1002/jps.2600610254.

(80) Shi, Y.; Luo, H. Q.; Li, N. B. Determination of the Critical Premicelle Concentration, First Critical Micelle Concentration and Second Critical Micelle

Concentration of Surfactants by Resonance Rayleigh Scattering Method without Any Probe. *Spectrochim. Acta A Mol. Biomol. Spectrosc.* **2011**, *78* (5). https://doi.org/10.1016/j.saa.2011.01.018.

(81) Ottaviani, G.; Wendelspiess, S.; Alvarez-Sánchez, R. Importance of Critical Micellar Concentration for the Prediction of Solubility Enhancement in Biorelevant Media. *Mol. Pharm.* **2015**, *12* (4). https://doi.org/10.1021/mp5006992.

(82) Evans, H. C. Alkyl Sulphates. Part i. Critical Micelle Concentrations of the Sodium Salts. *Journal of the Chemical Society (Resumed)* **1956**. https://doi.org/10.1039/jr9560000579.

(83) Burlatsky, S. F.; Atrazhev, V. V.; Dmitriev, D. V.; Sultanov, V. I.; Timokhina, E. N.; Ugolkova, E. A.; Tulyani, S.; Vincitore, A. Surface Tension Model for Surfactant Solutions at the Critical Micelle Concentration. *J. Colloid Interface Sci.* **2013**, *393* (1). https://doi.org/10.1016/j.jcis.2012.10.020.

(84) Wu, R.; Tian, M.; Shu, C.; Zhou, C.; Guan, W. Determination of the Critical Micelle Concentration of Surfactants Using Fluorescence Strategies. *Soft Matter*. 2022. https://doi.org/10.1039/d2sm01320g.

(85) Chen, J.; Hou, L.; Nan, J.; Ni, B.; Dai, W.; Ge, X. Prediction of Critical Micelle Concentration (CMC) of Surfactants Based on Structural Differentiation Using

Machine Learning. *Colloids Surf. A Physicochem. Eng. Asp.* **2024**, *703*. https://doi.org/10.1016/j.colsurfa.2024.135276.

(86) Huibers, P. D. T.; Lobanov, V. S.; Katritzky, A. R.; Shah, D. O.; Karelson, M. Prediction of Critical Micelle Concentration Using a Quantitative Structure-Property Relationship Approach. 1. Nonionic Surfactants. *Langmuir* **1996**, *12* (6). https://doi.org/10.1021/la950581j.

(87) Abooali, D.; Soleimani, R. Structure-Based Modeling of Critical Micelle Concentration (CMC) of Anionic Surfactants in Brine Using Intelligent Methods. *Sci. Rep.* **2023**, *13* (1). https://doi.org/10.1038/s41598-023-40466-1.

(88) Nivón-Ramírez, D.; Reyes-García, L. I.; Oviedo-Roa, R.; Gómez-Balderas, R.; Zuriaga-Monroy, C.; Martínez-Magadán, J. M. Critical Micelle Concentration of SDS through DPD Simulations Using COSMO-RS–Based Interaction Parameters, the Thermal Effects. *Colloids Surf. A Physicochem. Eng. Asp.* **2022**, *645*. https://doi.org/10.1016/j.colsurfa.2022.128867.

(89) Karimi, M. A.; Mozaheb, M. A.; Hatefi-Mehrjardi, A.; Tavallali, H.; Attaran, A. M.; Shamsi, R. A New Simple Method for Determining the Critical Micelle Concentration of Surfactants Using Surface Plasmon Resonance of Silver Nanoparticles. *J. Anal. Sci. Technol.* **2015**, *6* (1). https://doi.org/10.1186/s40543-015-0077-y.

(90) Rutkauskaite, A.; White, L. J.; Hilton, K. L. F.; Picci, G.; Croucher, L.; Caltagirone, C.; Hiscock, J. R. Supramolecular Self-Associating Amphiphiles: Determination of Molecular Self-Association Properties and Calculation of Critical Micelle Concentration Using a High-Throughput, Optical Density Based Methodology. *Org. Biomol. Chem.* **2022**, *20* (30). https://doi.org/10.1039/d2ob00066k.

(91) Mal, A.; Bag, S.; Ghosh, S.; Moulik, S. P. Physicochemistry of CTAB-SDS Interacted Catanionic Micelle-Vesicle Forming System: An Extended Exploration. *Colloids Surf. A Physicochem. Eng. Asp.* **2018**, *553*. https://doi.org/10.1016/j.colsurfa.2018.05.099.

(92) Seddon, D.; Müller, E. A.; Cabral, J. T. Machine Learning Hybrid Approach for the Prediction of Surface Tension Profiles of Hydrocarbon Surfactants in Aqueous Solution. *J. Colloid Interface Sci.* **2022**, *625*, 328–339. https://doi.org/10.1016/J.JCIS.2022.06.034.

(93) Brozos, C.; Rittig, J. G.; Bhattacharya, S.; Akanny, E.; Kohlmann, C.; Mitsos, A. Graph Neural Networks for Surfactant Multi-Property Prediction. *Colloids Surf. A Physicochem. Eng. Asp.* **2024**, *694*. https://doi.org/10.1016/j.colsurfa.2024.134133.

(94) Meissner, H. P.; Michaels, A. S. Surface Tensions of Pure Liquids and Liquid Mixtures. *Ind. Eng. Chem.* **1949**, *41* (12). https://doi.org/10.1021/ie50480a028.

(95) Nnadili, M.; Okafor, A. N.; Olayiwola, T.; Akinpelu, D.; Kumar, R.; Romagnoli, J. A. Surfactant-Specific AI-Driven Molecular Design: Integrating Generative Models, Predictive Modeling, and Reinforcement Learning for Tailored Surfactant Synthesis. *Ind. Eng. Chem. Res.* **2024**, *63* (14). https://doi.org/10.1021/acs.iecr.4c00401.

(96) González-Núñez, S.; Sumer, Z.; Amador, C.; Madhav, P.; Muralidharan, A.; Adjiman, C. S.; Martín, M. Design of Surfactant Molecules Under Performance Constraints. *ACS Sustain. Chem. Eng.* **2025**, *13* (34). https://doi.org/10.1021/acssuschemeng.5c04112.

(97) Banik, A.; Sengupta, U.; Hughes, H.; Sabhapathy, P.; Gurkan, B.; Pentzer, E. B.; Powers, D. C. Molecular Design Considerations for Azobenzene Anolytes. *ACS Omega* **2025**, *10* (24), 26199–26206. https://doi.org/10.1021/acsomega.5c05073.

(98) Tao, L.; Varshney, V.; Li, Y. Benchmarking Machine Learning Models for Polymer Informatics: An Example of Glass Transition Temperature. *Journal of Chemical Information and Modeling*. 2021. https://doi.org/10.1021/acs.jcim.1c01031.

(99) Galeazzo, T.; Shiraiwa, M. Predicting Glass Transition Temperature and Melting Point of Organic Compounds via Machine Learning and Molecular Embeddings. *Environmental Science: Atmospheres* **2022**. https://doi.org/10.1039/d1ea00090j.

(100) Mi, W.; Chen, H.; Zhu, D.; Zhang, T.; Qian, F. Melting Point Prediction of Organic Molecules by Deciphering the Chemical Structure into a Natural Language. *Chemical Communications*. 2021. https://doi.org/10.1039/d0cc07384a.

(101) Zhu, X.; Polyakov, V. R.; Bajjuri, K.; Hu, H.; Maderna, A.; Tovee, C. A.; Ward, S. C. Building Machine Learning Small Molecule Melting Points and Solubility Models Using CCDC Melting Points Dataset. *J. Chem. Inf. Model.* **2023**, *63* (10). https://doi.org/10.1021/acs.jcim.3c00308.

(102) Ran, Y.; He, Y.; Yang, G.; Johnson, J. L. H.; Yalkowsky, S. H. Estimation of Aqueous Solubility of Organic Compounds by Using the General Solubility Equation. *Chemosphere* **2002**, *48* (5). https://doi.org/10.1016/S0045-6535(02)00118-2.

(103) Hughes, L. D.; Palmer, D. S.; Nigsch, F.; Mitchell, J. B. O. Why Are Some Properties More Difficult to Predict than Others? A Study of QSPR Models of Solubility, Melting Point, and Log P. *J. Chem. Inf. Model.* **2008**, *48* (1). https://doi.org/10.1021/ci700307p.

(104) 'Krasnov, L.; 'Mikhaylov, S.; 'Fedorov, M.; 'Sosnin, S. *BigSolDB: Solubility Dataset of Compounds in Organic Solvents and Water in a Wide Range of Temperatures*; 2023.

(105) Pan, L.; Guo, Z.; Li, H.; Wang, Y.; Rao, H.; Jian, Q.; Sun, J.; Ren, J.; Wang, Z.; Liu, B.; Han, M.; Li, Y.; Fan, X.; Li, W.; Wei, L. High-Performance Porous Electrodes for Flow Batteries: Improvements of Specific Surface Areas and Reaction Kinetics. *ChemElectroChem*. 2024. https://doi.org/10.1002/celc.202400460.

(106) Zhang, L.; Shao, S. Image-Based Machine Learning for Materials Science. *Journal of Applied Physics*. 2022. https://doi.org/10.1063/5.0087381.

(107) Alldritt, B.; Hapala, P.; Oinonen, N.; Urtev, F.; Krejci, O.; Canova, F. F.; Kannala, J.; Schulz, F.; Liljeroth, P.; Foster, A. S. Automated Structure Discovery in Atomic Force Microscopy. *Sci. Adv.* **2020**, *6* (9). https://doi.org/10.1126/sciadv.aay6913.

(108) Li, W.; Field, K. G.; Morgan, D. Automated Defect Analysis in Electron Microscopic Images. *NPJ Comput. Mater.* **2018**, *4* (1). https://doi.org/10.1038/s41524-018-0093-8.

(109) Salthammer, T. Assessment of Methods for Predicting Physical and Chemical Properties of Organic Compounds. *Indoor Environments*. 2024. https://doi.org/10.1016/j.indenv.2024.100031.

(110) Cherkasov, A.; Muratov, E. N.; Fourches, D.; Varnek, A.; Baskin, I. I.; Cronin, M.; Dearden, J.; Gramatica, P.; Martin, Y. C.; Todeschini, R.; Consonni, V.; Kuz'Min, V. E.; Cramer, R.; Benigni, R.; Yang, C.; Rathman, J.; Terfloth, L.; Gasteiger, J.; Richard, A.; Tropsha, A. QSAR Modeling: Where Have You Been? Where Are You

Going To? *Journal of Medicinal Chemistry*. 2014. https://doi.org/10.1021/jm4004285.

(111) Liu, Y.; Kashima, H. Chemical Property Prediction under Experimental Biases. *Sci. Rep.* **2022**, *12* (1). https://doi.org/10.1038/s41598-022-12116-5.

(112) McClelland, H. E.; Jurs, P. C. Quantitative Structure-Property Relationships for the Prediction of Vapor Pressures of Organic Compounds from Molecular Structures. *J. Chem. Inf. Comput. Sci.* **2000**, *40* (4). https://doi.org/10.1021/ci990137c.

(113) Liu, F.; Cao, C.; Cheng, B. A Quantitative Structure-Property Relationship (QSPR) Study Of Aliphatic Alcohols by the Method of Dividing the Molecular Structure into Substructure. *Int. J. Mol. Sci.* **2011**, *12* (4). https://doi.org/10.3390/ijms12042448.

(114) Hechinger, M.; Leonhard, K.; Marquardt, W. What Is Wrong with Quantitative Structure-Property Relations Models Based on Three-Dimensional Descriptors? *J. Chem. Inf. Model.* **2012**, *52* (8). https://doi.org/10.1021/ci300246m.

(115) Rácz, A.; Bajusz, D.; Héberger, K. Intercorrelation Limits in Molecular Descriptor Preselection for QSAR/QSPR. *Mol. Inform.* **2019**, *38* (8–9). https://doi.org/10.1002/minf.201800154.

(116) Ng, W. Brief History of Electronic Structure Calculations in Computational Chemistry. *Journal of Nanotechnology Research* **2022**, *04* (01). https://doi.org/10.26502/jnr.2688-85210027.

(117) *Mathematical Challenges from Theoretical/Computational Chemistry*; 1995. https://doi.org/10.17226/4886.

(118) Becke, A. D. Perspective: Fifty Years of Density-Functional Theory in Chemical Physics. *Journal of Chemical Physics* **2014**, *140* (18). https://doi.org/10.1063/1.4869598.

(119) Tokatly, I. V. Time-Dependent Density Functional Theory for Many-Electron Systems Interacting with Cavity Photons. *Phys. Rev. Lett.* **2013**, *110* (23). https://doi.org/10.1103/PhysRevLett.110.233001.

(120) Lewin, M.; Paul, S. A Numerical Perspective on Hartree-Fock-Bogoliubov Theory. *ESAIM: Mathematical Modelling and Numerical Analysis*. 2014. https://doi.org/10.1051/m2an/2013094.

(121) Quantum Chemistry and Molecular Spectroscopy. *Choice Reviews Online* **1992**, *30* (03). https://doi.org/10.5860/choice.30-1527.

(122) Klamt, A. *COSMO-RS: From Quantum Chemistry to Fluid Phase Thermodynamics and Drug Design*; 2005.

(123) Abranches, D. O.; Maginn, E. J.; Colón, Y. J. Stochastic Machine Learning via Sigma Profiles to Build a Digital Chemical Space. *Proc. Natl. Acad. Sci. U. S. A.* **2024**, *121* (31). https://doi.org/10.1073/pnas.2404676121.

(124) Salih, F. Y. M.; Abranches, D. O.; Maginn, E. J.; Colón, Y. J. Open-Source Generation of Sigma Profiles: Impact of Quantum Chemistry and Solvation Treatment on Machine Learning Performance. *Digital Discovery* **2025**, *4* (10). https://doi.org/10.1039/d5dd00087d.

(125) Lahouari, A.; Rogal, J.; Tuckerman, M. E. Automated Machine Learning Pipeline: Large Language Models-Assisted Automated Data Set Generation for Training Machine-Learned Interatomic Potentials. *J. Chem. Theory Comput.* **2026**, *22* (1). https://doi.org/10.1021/acs.jctc.5c01610.

(126) Chen, M. S.; Robledo, A.; Schäfer, C.; Han, K. Y.; Clementi, C.; Tuckerman, M. E. Machine Learning-Accelerated Path Integral Molecular Dynamics Simulations of Reactive Organic Electrolytes. *Journal of Chemical Physics* **2025**, *163* (14). https://doi.org/10.1063/5.0288380.

(127) Fradkov, A. L. Early History of Machine Learning. In *IFAC-PapersOnLine*; 2020; Vol. 53. https://doi.org/10.1016/j.ifacol.2020.12.1888.

(128) Pengelley, D. J.; Simmons, G. F. Calculus Gems: Brief Lives and Memorable Mathematics. *The American Mathematical Monthly* **1994**, *101* (4). https://doi.org/10.2307/2975640.

(129) Topping, J. The Works of Archimedes. *Physics Bulletin* **1953**, *4* (48). https://doi.org/10.1088/0031-9112/4/48/004.

(130) Jacobi, C. G. J. De Determinantibus Functionalibus. In *C. G. J. Jacobi's Gesammelte Werke*; 2014. https://doi.org/10.1017/cbo9781139567978.015.

(131) Jacobi, C. G. J. 12. De Determinantibus Functionalibus. *Journal fur die Reine und Angewandte Mathematik* **1841**, *1841* (22). https://doi.org/10.1515/crll.1841.22.319.

(132) Ewing, J.; Kolmogorov, A. N.; Yushkevich, A. P.; Cooke, R. Mathematics of the 19th Century: Geometry/Analytic Function Theory. *The College Mathematics Journal* **1999**, *30* (2). https://doi.org/10.2307/2687730.

(133) Curry, H. B. The Method of Steepest Descent for Non-Linear Minimization Problems. *Q. Appl. Math.* **1944**, *2* (3), 258–261. https://doi.org/10.1090/qam/10667.

(134) Lemaréchal, C. Cauchy and the Gradient Method. In *Optimization Stories*; 2023. https://doi.org/10.4171/dms/6/27.

(135) Rosenblatt, F. The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain. *Psychol. Rev.* **1958**, *65* (6), 386–408. https://doi.org/10.1037/h0042519.

(136) Ivakhnenko, A. G.; Lapa, V. . G. *Cybernetic Predicting Devices*; 1966.

(137) Ivakhnenko, A. G.; Lapa, V. G.; Valentin Grigor′evich Lapa. Cybernetics_and_Forecasting_Techniques. *American Elsevier Publishing Company* **1967**.

(138) Rosenblatt, F. Perceptron Simulation Experiments. *Proceedings of the IRE* **1960**, *48* (3), 301–309. https://doi.org/10.1109/JRPROC.1960.287598.

(139) Rosenblatt, F. *The Perceptron—a Perceiving and Recognizing Automaton*; 1957.

(140) Tsantekidis, A.; Passalis, N.; Tefas, A. Recurrent Neural Networks. In *Deep Learning for Robot Perception and Cognition*; 2022. https://doi.org/10.1016/B978-0-32-385787-1.00010-5.

(141) Liu, G.; Furth, N.; Shi, H.; Khreishah, A.; Lee, J. Y.; Ansari, N.; Liu, C.; Jararweh, Y. Federated Learning Aided Deep Convolutional Neural Network Solution for Smart Traffic Management. In *Proceedings of IEEE/IFIP Network Operations and Management Symposium 2023, NOMS 2023*; 2023. https://doi.org/10.1109/NOMS56928.2023.10154300.

(142) Fukushima, K. Neocognitron: A Self-Organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position. *Biol. Cybern.* **1980**, *36* (4). https://doi.org/10.1007/BF00344251.

(143) Schmidhuber, J. Annotated History of Modern AI and Deep Learning. **2022**. https://doi.org/10.48550/arXiv.2212.11279.

(144) Rumelhart, D. E.; Hinton, G. E.; Williams, R. J. Learning Internal Representations by Error Propagation. In *Readings in Cognitive Science: A Perspective from Psychology and Artificial Intelligence*; 2013. https://doi.org/10.1016/B978-1-4832-1446-7.50035-2.

(145) Toosi, A.; Bottino, A. G.; Saboury, B.; Siegel, E.; Rahmim, A. A Brief History of AI: How to Prevent Another Winter (A Critical Review). *PET Clinics*. 2021. https://doi.org/10.1016/j.cpet.2021.07.001.

(146) Zhong, X.; Gallagher, B.; Liu, S.; Kailkhura, B.; Hiszpanski, A.; Han, T. Y. J. Explainable Machine Learning in Materials Science. *npj Computational Materials*. 2022. https://doi.org/10.1038/s41524-022-00884-7.

(147) Packwood, D.; Nguyen, L. T. H.; Cesana, P.; Zhang, G.; Staykov, A.; Fukumoto, Y.; Nguyen, D. H. Machine Learning in Materials Chemistry: An Invitation. *Machine Learning with Applications* **2022**, *8*, 100265. https://doi.org/10.1016/J.MLWA.2022.100265.

(148) Deringer, V. L.; Bartók, A. P.; Bernstein, N.; Wilkins, D. M.; Ceriotti, M.; Csányi, G. Gaussian Process Regression for Materials and Molecules. *Chemical Reviews*. 2021. https://doi.org/10.1021/acs.chemrev.1c00022.

(149) Cai, J.; Chu, X.; Xu, K.; Li, H.; Wei, J. Machine Learning-Driven New Material Discovery. *Nanoscale Advances*. 2020. https://doi.org/10.1039/d0na00388c.

(150) Liu, Y.; Zhao, T.; Ju, W.; Shi, S. Materials Discovery and Design Using Machine Learning. *Journal of Materiomics*. 2017. https://doi.org/10.1016/j.jmat.2017.08.002.

(151) Chan, C. H.; Sun, M.; Huang, B. Application of Machine Learning for Advanced Material Prediction and Design. *EcoMat* **2022**, *4* (4). https://doi.org/10.1002/eom2.12194.

(152) Fang, J.; Xie, M.; He, X.; Zhang, J.; Hu, J.; Chen, Y.; Yang, Y.; Jin, Q. Machine Learning Accelerates the Materials Discovery. *Materials Today Communications*. 2022. https://doi.org/10.1016/j.mtcomm.2022.104900.

(153) Liu, Y.; Guo, B.; Zou, X.; Li, Y.; Shi, S. Machine Learning Assisted Materials Design and Discovery for Rechargeable Batteries. *Energy Storage Materials*. 2020. https://doi.org/10.1016/j.ensm.2020.06.033.

(154) Schmidt, J.; Marques, M. R. G.; Botti, S.; Marques, M. A. L. Recent Advances and Applications of Machine Learning in Solid-State Materials Science. *npj Computational Materials*. 2019. https://doi.org/10.1038/s41524-019-0221-0.

(155) Wei, Z.; He, Q.; Zhao, Y. Machine Learning for Battery Research. *J. Power Sources* **2022**, *549*. https://doi.org/10.1016/j.jpowsour.2022.232125.

(156) Dybowski, R. Interpretable Machine Learning as a Tool for Scientific Discovery in Chemistry. *New Journal of Chemistry*. 2020. https://doi.org/10.1039/d0nj02592e.

(157) Keith, J. A.; Vassilev-Galindo, V.; Cheng, B.; Chmiela, S.; Gastegger, M.; Müller, K. R.; Tkatchenko, A. Combining Machine Learning and Computational Chemistry for Predictive Insights into Chemical Systems. *Chemical Reviews*. 2021. https://doi.org/10.1021/acs.chemrev.1c00107.

(158) Campbell, M.; Hoane, A. J.; Hsu, F. H. Deep Blue. *Artif. Intell.* **2002**, *134* (1–2). https://doi.org/10.1016/S0004-3702(01)00129-1.

(159) Vapnik, V. N.; Chervonenkis, A. Ya. A Class of Algorithms for Pattern Recognition Learning. *Automatika i Telemekhanika* **1964**, *25* (6).

(160) Rasmussen, C. E. Gaussian Processes in Machine Learning. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* **2004**, *3176*. https://doi.org/10.1007/978-3-540-28650-9_4.

(161) Rasmussen, C. E.; Nickisch, H. Gaussian Processes for Machine Learning (GPML) Toolbox. *Journal of Machine Learning Research* **2010**, *11*.

(162) Scarselli, F.; Gori, M.; Tsoi, A. C.; Hagenbuchner, M.; Monfardini, G. The Graph Neural Network Model. *IEEE Trans. Neural Netw.* **2009**, *20* (1). https://doi.org/10.1109/TNN.2008.2005605.

(163) Hammond, D. K.; Vandergheynst, P.; Gribonval, R. Wavelets on Graphs via Spectral Graph Theory. arXiv 2009. https://doi.org/10.48550/ARXIV.0912.3848.

(164) Sun, M.; Wu, T.; Xue, Y.; Dougherty, A. W.; Huang, B.; Li, Y.; Yan, C. H. Mapping of Atomic Catalyst on Graphdiyne. *Nano Energy* **2019**, *62*. https://doi.org/10.1016/j.nanoen.2019.06.008.

(165) Coley, C. W.; Barzilay, R.; Green, W. H.; Jaakkola, T. S.; Jensen, K. F. Convolutional Embedding of Attributed Molecular Graphs for Physical Property Prediction. *J. Chem. Inf. Model.* **2017**, *57* (8). https://doi.org/10.1021/acs.jcim.6b00601.

(166) Liu, Q.; Allamanis, M.; Brockschmidt, M.; Gaunt, A. L. Constrained Graph Variational Autoencoders for Molecule Design. In *Advances in Neural Information Processing Systems*; 2018; Vol. 2018-December.

(167) Kim, Y.; Jeong, Y.; Kim, J.; Lee, E. K.; Kim, W. J.; Choi, I. S. MolNet: A Chemically Intuitive Graph Neural Network for Prediction of Molecular Properties. arXiv 2022. https://doi.org/10.48550/ARXIV.2203.09456.

(168) Chen, C.; Ye, W.; Zuo, Y.; Zheng, C.; Ong, S. P. Graph Networks as a Universal Machine Learning Framework for Molecules and Crystals. *Chemistry of Materials* **2019**, *31* (9), 3564–3572. https://doi.org/10.1021/acs.chemmater.9b01294.

(169) Park, C. W.; Wolverton, C. Developing an Improved Crystal Graph Convolutional Neural Network Framework for Accelerated Materials Discovery. *Phys. Rev. Mater.* **2020**, *4* (6). https://doi.org/10.1103/PhysRevMaterials.4.063801.

(170) Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; Bengio, Y. Generative Adversarial Networks. *Commun. ACM* **2020**, *63* (11). https://doi.org/10.1145/3422622.

(171) Gui, J.; Sun, Z.; Wen, Y.; Tao, D.; Ye, J. A Review on Generative Adversarial Networks: Algorithms, Theory, and Applications. *IEEE Trans. Knowl. Data Eng.* **2023**, *35* (4). https://doi.org/10.1109/TKDE.2021.3130191.

(172) Weng, Y.; Zhou, T.; Li, Y.; Qiu, X. NAS-Unet: Neural Architecture Search for Medical Image Segmentation. *IEEE Access* **2019**, *7*. https://doi.org/10.1109/ACCESS.2019.2908991.

(173) Krizhevsky, A.; Sutskever, I.; Hinton, G. E. ImageNet Classification with Deep Convolutional Neural Networks. *Commun. ACM* **2017**, *60* (6). https://doi.org/10.1145/3065386.

(174) Kalinin, S. V. (Keynote) Active Learning of Structure-Dependent Electrochemical Reactivity on the Nanometer Scale with the ML-Enabled Electrochemical Strain Microscopy . *ECS Meeting Abstracts* **2025**, *MA2025-02* (69). https://doi.org/10.1149/ma2025-02693321mtgabs.

(175) Kalinin, S. V.; Ziatdinov, M.; Hinkle, J.; Jesse, S.; Ghosh, A.; Kelley, K. P.; Lupini, A. R.; Sumpter, B. G.; Vasudevan, R. K. Automated and Autonomous Experiments in Electron and Scanning Probe Microscopy. *ACS Nano*. 2021. https://doi.org/10.1021/acsnano.1c02104.

(176) Channi, H. K.; Kumar, P.; Sandhu, R.; Syam, F. A. Energy Storage Using Computer Vision: Control and Optimization of Energy Storage. In *Computer Vision and Machine Intelligence for Renewable Energy Systems*; 2024. https://doi.org/10.1016/B978-0-443-28947-7.00012-4.

(177) Kanagaraj, N.; Hicks, D.; Goyal, A.; Tiwari, S.; Singh, G. Deep Learning Using Computer Vision in Self Driving Cars for Lane and Traffic Sign Detection. *International Journal of System Assurance Engineering and Management* **2021**, *12* (6). https://doi.org/10.1007/s13198-021-01127-6.

(178) Li, L.; Mu, X.; Li, S.; Peng, H. A Review of Face Recognition Technology. *IEEE Access* **2020**, *8*. https://doi.org/10.1109/ACCESS.2020.3011028.

(179) Wilson, A. G.; Hu, Z.; Salakhutdinov, R.; Xing, E. P. *Deep Kernel Learning*; 2015.

(180) Chen, T.; Guestrin, C. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*; KDD '16; Association for Computing Machinery: New York, NY, USA, 2016; pp 785–794. https://doi.org/10.1145/2939672.2939785.

(181) OpenAI. *Introducing GPT-5*. OpenAI.

(182) OpenAI; :; Achiam, J.; Adler, S.; Agarwal, S.; Ahmad, L.; Akkaya, I.; Aleman, F. L.; Almeida, D.; Altenschmidt, J.; Altman, S.; Anadkat, S.; Avila, R.; Babuschkin, I.; Balaji, S.; Balcom, V.; Baltescu, P.; Bao, H.; Bavarian, M.; Belgum, J.; Bello, I.; Berdine, J.; Bernadett-Shapiro, G.; Berner, C.; Bogdonoff, L.; Boiko, O.; Boyd, M.; Brakman, A.-L.; Brockman, G.; Brooks, T.; Brundage, M.; Button, K.; Cai, T.; Campbell, R.; Cann, A.; Carey, B.; Carlson, C.; Carmichael, R.; Chan, B.; Chang, C.; Chantzis, F.; Chen, D.; Chen, S.; Chen, R.; Chen, J.; Chen, M.; Chess, B.; Cho, C.; Chu, C.; Chung, H. W.; Cummings, D.; Currier, J.; Dai, Y.; Decareaux, C.; Degry, T.; Deutsch, N.; Deville, D.; Dhar, A.; Dohan, D.; Dowling, S.; Dunning, S.; Ecoffet, A.; Eleti, A.; Eloundou, T.; Farhi, D.; Fedus, L.; Felix, N.; Fishman, S. P.; Forte, J.; Fulford, I.; Gao, L.; Georges, E.; Gibson, C.; Goel, V.; Gogineni, T.; Goh, G.; Gontijo-Lopes, R.; Gordon, J.; Grafstein, M.; Gray, S.; Greene, R.; Gross, J.;

Gu, S. S.; Guo, Y.; Hallacy, C.; Han, J.; Harris, J.; He, Y.; Heaton, M.; Heidecke, J.; Hesse, C.; Hickey, A.; Hickey, W.; Hoeschele, P.; Houghton, B.; Hsu, K.; Hu, S.; Hu, X.; Huizinga, J.; Jain, S.; Jain, S.; Jang, J.; Jiang, A.; Jiang, R.; Jin, H.; Jin, D.; Jomoto, S.; Jonn, B.; Jun, H.; Kaftan, T.; Kaiser, Ł.; Kamali, A.; Kanitscheider, I.; Keskar, N. S.; Khan, T.; Kilpatrick, L.; Kim, J. W.; Kim, C.; Kim, Y.; Kirchner, H.; Kiros, J.; Knight, M.; Kokotajlo, D.; Kondraciuk, Ł.; Kondrich, A.; Konstantinidis, A.; Kosic, K.; Krueger, G.; Kuo, V.; Lampe, M.; Lan, I.; Lee, T.; Leike, J.; Leung, J.; Levy, D.; Li, C. M.; Lim, R.; Lin, M.; Lin, S.; Litwin, M.; Lopez, T.; Lowe, R.; Lue, P.; Makanju, A.; Malfacini, K.; Manning, S.; Markov, T.; Markovski, Y.; Martin, B.; Mayer, K.; Mayne, A.; McGrew, B.; McKinney, S. M.; McLeavey, C.; McMillan, P.; McNeil, J.; Medina, D.; Mehta, A.; Menick, J.; Metz, L.; Mishchenko, A.; Mishkin, P.; Monaco, V.; Morikawa, E.; Mossing, D.; Mu, T.; Murati, M.; Murk, O.; Mély, D.; Nair, A.; Nakano, R.; Nayak, R.; Neelakantan, A.; Ngo, R.; Noh, H.; Ouyang, L.; O'Keefe, C.; Pachocki, J.; Paino, A.; Palermo, J.; Pantuliano, A.; Parascandolo, G.; Parish, J.; Parparita, E.; Passos, A.; Pavlov, M.; Peng, A.; Perelman, A.; Peres, F. de A. B.; Petrov, M.; Pinto, H. P. de O.; Michael; Pokorny; Pokrass, M.; Pong, V.; Powell, T.; Power, A.; Power, B.; Proehl, E.; Puri, R.; Radford, A.; Rae, J.; Ramesh, A.; Raymond, C.; Real, F.; Rimbach, K.; Ross, C.; Rotsted, B.; Roussez, H.; Ryder, N.; Saltarelli, M.; Sanders, T.; Santurkar, S.; Sastry, G.; Schmidt, H.; Schnurr, D.; Schulman, J.; Selsam, D.; Sheppard, K.; Sherbakov, T.; Shieh, J.; Shoker, S.; Shyam, P.; Sidor, S.; Sigler, E.; Simens, M.;

Sitkin, J.; Slama, K.; Sohl, I.; Sokolowsky, B.; Song, Y.; Staudacher, N.; Such, F. P.; Summers, N.; Sutskever, I.; Tang, J.; Tezak, N.; Thompson, M.; Tillet, P.; Tootoonchian, A.; Tseng, E.; Tuggle, P.; Turley, N.; Tworek, J.; Uribe, J. F. C.; Vallone, A.; Vijayvergiya, A.; Voss, C.; Wainwright, C.; Wang, J. J.; Wang, A.; Wang, B.; Ward, J.; Wei, J.; Weinmann, C.; Welihinda, A.; Welinder, P.; Weng, J.; Weng, L.; Wiethoff, M.; Willner, D.; Winter, C.; Wolrich, S.; Wong, H.; Workman, L.; Wu, S.; Wu, J.; Wu, M.; Xiao, K.; Xu, T.; Yoo, S.; Yu, K.; Yuan, Q.; Zaremba, W.; Zellers, R.; Zhang, C.; Zhang, M.; Zhao, S.; Zheng, T.; Zhuang, J.; Zhuk, W.; Zoph, B. GPT-4 Technical Report. **2023**.

(183) Bachstein, M.; Gropp, C.; Hazlewood, V.; Peterson, G. Lessons from Benchmarking the AI Tennessee Initiative Resources. In *PEARC 2024 - Practice and Experience in Advanced Research Computing 2024: Human Powered Computing*; 2024. https://doi.org/10.1145/3626203.3670524.

(184) Wilkins, M.; Wang, H.; Liu, P.; Pham, B.; Guo, Y.; Thakur, R.; Dinda, P.; Hardavellas, N. Generalized Collective Algorithms for the Exascale Era. In *Proceedings - IEEE International Conference on Cluster Computing, ICCC*; 2023. https://doi.org/10.1109/CLUSTER52292.2023.00013.

(185) McCorkle, M. *ORNL Launches Summit Supercomputer*. The U.S. Department of Energy’s Oak Ridge National Laboratory.

(186) Nahar, N.; Zhang, H.; Lewis, G.; Zhou, S.; Kästner, C. The Product Beyond the Model - An Empirical Study of Repositories of Open-Source ML Products. In *Proceedings - International Conference on Software Engineering*; 2025. https://doi.org/10.1109/ICSE55347.2025.00006.

(187) Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; Desmaison, A.; Köpf, A.; Yang, E.; DeVito, Z.; Raison, M.; Tejani, A.; Chilamkurthy, S.; Steiner, B.; Fang, L.; Bai, J.; Chintala, S. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems*; 2019; Vol. 32.

(188) Ketkar, N.; Moolayil, J. *Deep Learning with Python*; 2021. https://doi.org/10.1007/978-1-4842-5364-9.

(189) Abadi, M.; Barham, P.; Chen, J.; Chen, Z.; Davis, A.; Dean, J.; Devin, M.; Ghemawat, S.; Irving, G.; Isard, M.; Kudlur, M.; Levenberg, J.; Monga, R.; Moore, S.; Murray, D. G.; Steiner, B.; Tucker, P.; Vasudevan, V.; Warden, P.; Wicke, M.; Yu, Y.; Zheng, X. TensorFlow: A System for Large-Scale Machine Learning. In *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2016*; 2016.

(190) Hughes, L. D.; Palmer, D. S.; Nigsch, F.; Mitchell, J. B. O. Why Are Some Properties More Difficult to Predict than Others? A Study of QSPR Models of

Solubility, Melting Point, and Log P. *J. Chem. Inf. Model.* **2008**, *48* (1). https://doi.org/10.1021/ci700307p.

(191) Van Tilborg, D.; Alenicheva, A.; Grisoni, F. Exposing the Limitations of Molecular Machine Learning with Activity Cliffs. *J. Chem. Inf. Model.* **2022**, *62* (23). https://doi.org/10.1021/acs.jcim.2c01073.

(192) Chen, Z.; Li, D.; Liu, M.; Liu, J. Graph Neural Networks with Molecular Segmentation for Property Prediction and Structure–Property Relationship Discovery. *Comput. Chem. Eng.* **2023**, *179*. https://doi.org/10.1016/j.compchemeng.2023.108403.

(193) Wu, Z.; Wang, J.; Du, H.; Jiang, D.; Kang, Y.; Li, D.; Pan, P.; Deng, Y.; Cao, D.; Hsieh, C. Y.; Hou, T. Chemistry-Intuitive Explanation of Graph Neural Networks for Molecular Property Prediction with Substructure Masking. *Nat. Commun.* **2023**, *14* (1). https://doi.org/10.1038/s41467-023-38192-3.

(194) Thakkar, P.; Khatri, S.; Dobariya, D.; Patel, D.; Dey, B.; Singh, A. K. Advances in Materials and Machine Learning Techniques for Energy Storage Devices: A Comprehensive Review. *Journal of Energy Storage*. 2024. https://doi.org/10.1016/j.est.2024.110452.

(195) Nandy, A.; Duan, C.; Kulik, H. J. Audacity of Huge: Overcoming Challenges of Data Scarcity and Data Quality for Machine Learning in Computational Materials

Discovery. *Current Opinion in Chemical Engineering*. 2022. https://doi.org/10.1016/j.coche.2021.100778.

(196) Ralchenko, Y.; Kramida, A. Development of NIST Atomic Databases and Online Tools. *Atoms* **2020**, *8* (3). https://doi.org/10.3390/ATOMS8030056.

(197) Kim, S.; Chen, J.; Cheng, T.; Gindulyte, A.; He, J.; He, S.; Li, Q.; Shoemaker, B. A.; Thiessen, P. A.; Yu, B.; Zaslavsky, L.; Zhang, J.; Bolton, E. E. PubChem 2023 Update. *Nucleic Acids Res.* **2023**, *51* (D1), D1373–D1380. https://doi.org/10.1093/nar/gkac956.

(198) Ramakrishnan, R.; Dral, P. O.; Rupp, M.; Von Lilienfeld, O. A. Quantum Chemistry Structures and Properties of 134 Kilo Molecules. *Sci. Data* **2014**, *1*. https://doi.org/10.1038/sdata.2014.22.

(199) Irwin, J. J.; Shoichet, B. K. ZINC - A Free Database of Commercially Available Compounds for Virtual Screening. *J. Chem. Inf. Model.* **2005**, *45* (1). https://doi.org/10.1021/ci049714+.

(200) Kirklin, S.; Saal, J. E.; Meredig, B.; Thompson, A.; Doak, J. W.; Aykol, M.; Rühl, S.; Wolverton, C. The Open Quantum Materials Database (OQMD): Assessing the Accuracy of DFT Formation Energies. *NPJ Comput. Mater.* **2015**, *1*. https://doi.org/10.1038/npjcompumats.2015.10.

(201) Landis, D. D.; Hummelshøj, J. S.; Nestorov, S.; Greeley, J.; Dułłak, M.; Bligaard, T.; Nørskov, J. K.; Jacobsen, K. W. The Computational Materials Repository. *Comput. Sci. Eng.* **2012**, *14* (6). https://doi.org/10.1109/MCSE.2012.16.

(202) Allmann, R.; Hinek, R. The Introduction of Structure Types into the Inorganic Crystal Structure Database ICSD. *Acta Crystallogr. A* **2007**, *63* (5). https://doi.org/10.1107/S0108767307038081.

(203) Graulis, S.; Chateigner, D.; Downs, R. T.; Yokochi, A. F. T.; Quirós, M.; Lutterotti, L.; Manakova, E.; Butkus, J.; Moeck, P.; Le Bail, A. Crystallography Open Database - An Open-Access Collection of Crystal Structures. *J. Appl. Crystallogr.* **2009**, *42* (4). https://doi.org/10.1107/S0021889809016690.

(204) Royal Society of Chemistry. ChemSpider. Search and Share Chemistry. *Royal Society of Chemistry*. 2015.

(205) Groom, C. R.; Bruno, I. J.; Lightfoot, M. P.; Ward, S. C. The Cambridge Structural Database. *Acta Crystallogr. B Struct. Sci. Cryst. Eng. Mater.* **2016**, *72* (2). https://doi.org/10.1107/S2052520616003954.

(206) Sushko, I.; Novotarskyi, S.; Körner, R.; Pandey, A. K.; Rupp, M.; Teetz, W.; Brandmaier, S.; Abdelaziz, A.; Prokopenko, V. V.; Tanchuk, V. Y.; Todeschini, R.; Varnek, A.; Marcou, G.; Ertl, P.; Potemkin, V.; Grishina, M.; Gasteiger, J.; Schwab, C.; Baskin, I. I.; Palyulin, V. A.; Radchenko, E. V.; Welsh, W. J.; Kholodovych, V.;

Chekmarev, D.; Cherkasov, A.; Aires-De-Sousa, J.; Zhang, Q. Y.; Bender, A.; Nigsch, F.; Patiny, L.; Williams, A.; Tkachenko, V.; Tetko, I. V. Online Chemical Modeling Environment (OCHEM): Web Platform for Data Storage, Model Development and Publishing of Chemical Information. *J. Comput. Aided. Mol. Des.* **2011**, *25* (6). https://doi.org/10.1007/s10822-011-9440-2.

(207) Borysov, S. S.; Geilhufe, R. M.; Balatsky, A. V. Organic Materials Database: An Open-Access Online Database for Data Mining. *PLoS One* **2017**, *12* (2). https://doi.org/10.1371/journal.pone.0171501.

(208) Downs, R. T.; Hall-Wallace, M. The American Mineralogist Crystal Structure Database. *American Mineralogist* **2003**, *88* (1). https://doi.org/10.5860/choice.43sup-0302.

(209) Hjorth Larsen, A.; JØrgen Mortensen, J.; Blomqvist, J.; Castelli, I. E.; Christensen, R.; Dułak, M.; Friis, J.; Groves, M. N.; Hammer, B.; Hargus, C.; Hermes, E. D.; Jennings, P. C.; Bjerre Jensen, P.; Kermode, J.; Kitchin, J. R.; Leonhard Kolsbjerg, E.; Kubal, J.; Kaasbjerg, K.; Lysgaard, S.; Bergmann Maronsson, J.; Maxson, T.; Olsen, T.; Pastewka, L.; Peterson, A.; Rostgaard, C.; SchiØtz, J.; Schütt, O.; Strange, M.; Thygesen, K. S.; Vegge, T.; Vilhelmsen, L.; Walter, M.; Zeng, Z.; Jacobsen, K. W. The Atomic Simulation Environment - A Python Library for Working with Atoms. *Journal of Physics Condensed Matter*. 2017. https://doi.org/10.1088/1361-648X/aa680e.

(210) Madsen, J.; Susi, T. The AbTEM Code: Transmission Electron Microscopy from First Principles. *Open Research Europe* **2021**, *1*. https://doi.org/10.12688/openreseurope.13015.2.

(211) Ghosh, A.; Kalinin, S. V.; Ziatdinov, M. A. Discovery of Structure–Property Relations for Molecules via Hypothesis-Driven Active Learning over the Chemical Space. *APL Machine Learning* **2023**, *1* (4). https://doi.org/10.1063/5.0157644.

(212) Klamt, A.; Jonas, V.; Bürger, T.; Lohrenz, J. C. W. Refinement and Parametrization of COSMO-RS. *Journal of Physical Chemistry A* **1998**, *102* (26). https://doi.org/10.1021/jp980017s.

(213) NCI/Cadd Chemical Identifier Resolver. *National Institutes of Health*. U.S. Department of Health and Human Services. https://cactus.nci.nih.gov/chemical/structure.

(214) Brown, T. ChemDraw. *The Science Teacher* **2014**, *81* (2).

(215) *RDKit: Open-Source cheminformatics.* . https://www.rdkit.org https://doi.org/10.5281/zenodo.591637 (accessed 2023-03-02).

(216) Himanen, L.; Jäger, M. O. J.; Morooka, E. V.; Federici Canova, F.; Ranawat, Y. S.; Gao, D. Z.; Rinke, P.; Foster, A. S. DScribe: Library of Descriptors for Machine Learning in Materials Science. *Comput. Phys. Commun.* **2020**, *247*, 106949. https://doi.org/10.1016/J.CPC.2019.106949.

(217) Yap, C. W. PaDEL-Descriptor: An Open Source Software to Calculate Molecular Descriptors and Fingerprints. *J. Comput. Chem.* **2011**, *32* (7). https://doi.org/10.1002/jcc.21707.

(218) Rogers, D.; Hahn, M. Extended-Connectivity Fingerprints. *J. Chem. Inf. Model.* **2010**, *50* (5). https://doi.org/10.1021/ci100050t.

(219) Bajorath, J.; Jasial, S.; Hu, Y.; Vogt, M. Activity-Relevant Similarity Values for Fingerprints and Implications for Similarity Searching. *F1000Res.* **2016**, *5*. https://doi.org/10.12688/f1000research.8357.1.

(220) Coupry, D. E.; Pogány, P. Application of Deep Metric Learning to Molecular Graph Similarity. *J. Cheminform.* **2022**, *14* (1). https://doi.org/10.1186/s13321-022-00595-7.

(221) Simonovsky, M.; Komodakis, N. GraphVAE: Towards Generation of Small Graphs Using Variational Autoencoders. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*; 2018; Vol. 11139 LNCS. https://doi.org/10.1007/978-3-030-01418-6_41.

(222) Kipf Thomas N; Welling, M. Variational Graph Auto-Encoders. November 21, 2016.

(223) Shi, X.; Lv, F.; Seng, D.; Zhang, J.; Chen, J.; Xing, B. Visualizing and Understanding Graph Convolutional Network. *Multimed. Tools Appl.* **2021**, *80* (6). https://doi.org/10.1007/s11042-020-09885-4.

(224) Yang, X.; Yan, M.; Pan, S.; Ye, X.; Fan, D. Simple and Efficient Heterogeneous Graph Neural Network. arXiv 2022. https://doi.org/10.48550/ARXIV.2207.02547.

(225) Fung, V.; Zhang, J.; Juarez, E.; Sumpter, B. G. Benchmarking Graph Neural Networks for Materials Chemistry. *NPJ Comput. Mater.* **2021**, *7* (1), 84.

(226) Reiser, P.; Neubert, M.; Eberhard, A.; Torresi, L.; Zhou, C.; Shao, C.; Metni, H.; van Hoesel, C.; Schopmans, H.; Sommer, T.; Friederich, P. Graph Neural Networks for Materials Science and Chemistry. arXiv 2022. https://doi.org/10.48550/ARXIV.2208.09481.

(227) Kipf, T. N.; Welling, M. Semi-Supervised Classification with Graph Convolutional Networks. arXiv 2016. https://doi.org/10.48550/ARXIV.1609.02907.

(228) Zhou, J.; Cui, G.; Hu, S.; Zhang, Z.; Yang, C.; Liu, Z.; Wang, L.; Li, C.; Sun, M. Graph Neural Networks: A Review of Methods and Applications. *AI Open* **2020**, *1*, 57–81. https://doi.org/https://doi.org/10.1016/j.aiopen.2021.01.001.

(229) Louis, S. Y.; Zhao, Y.; Nasiri, A.; Wang, X.; Song, Y.; Liu, F.; Hu, J. Graph Convolutional Neural Networks with Global Attention for Improved Materials

Property Prediction. *Physical Chemistry Chemical Physics* **2020**, *22* (32). https://doi.org/10.1039/d0cp01474e.

(230) Ye, S.; Liang, J.; Liu, R.; Zhu, X. Symmetrical Graph Neural Network for Quantum Chemistry with Dual Real and Momenta Space. *J. Phys. Chem. A* **2020**, *124* (34), 6945–6953. https://doi.org/10.1021/acs.jpca.0c03201.

(231) Xie, T.; Grossman, J. C. Crystal Graph Convolutional Neural Networks for an Accurate and Interpretable Prediction of Material Properties. *Phys. Rev. Lett.* **2018**, *120* (14). https://doi.org/10.1103/physrevlett.120.145301.

(232) Wang, J.; Wolf, R. M.; Caldwell, J. W.; Kollman, P. A.; Case, D. A. Development and Testing of a General Amber Force Field. *J Comput Chem* **2004**, *25* (9), 1157–1174.

(233) Halgren, T. A. Merck Molecular Force Field. I. Basis, Form, Scope, Parameterization, and Performance of MMFF94. *J. Comput. Chem.* **1996**, *17* (5-6), 490–641. https://doi.org/10.1002/(sici)1096-987x(199604)17:5/6<490::aid-jcc1>3.0.co;2-p.

(234) Rupp, M.; Tkatchenko, A.; Müller, K.-R.; von Lilienfeld, O. A. Fast and Accurate Modeling of Molecular Atomization Energies with Machine Learning. *Phys. Rev. Lett.* **2012**, *108* (5), 058301. https://doi.org/10.1103/PhysRevLett.108.058301.

(235) Schmidhuber, J. Deep Learning in Neural Networks: An Overview. *Neural Networks*. 2015. https://doi.org/10.1016/j.neunet.2014.09.003.

(236) Kaiming, H.; Xiangyu, Z.; Shaoqing, R.; Jian, S. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification Kaiming. *Biochem. Biophys. Res. Commun.* **2018**, *498* (1).

(237) Ucar, K. Improving Electric Vehicle State of Charge Estimation with Wavelet Transform-Integrated 1D-CNN Pooling Layers. *J. Energy Storage* **2025**, *117*. https://doi.org/10.1016/j.est.2025.116202.

(238) Baeldung. Dropout Layer. *Https://Www.Baeldung.Com/Cs/Ml-Relu-Dropout-Layers*. 2023.

(239) Gilmer, J.; Schoenholz, S. S.; Riley, P. F.; Vinyals, O.; Dahl, G. E. Neural Message Passing for Quantum Chemistry. *CoRR* **2017**, *abs/1704.01212*.

(240) Begum, R.; Mollah, A. F. Convolutional Autoencoder-Based Models for Image Denoising: A Comparative Study; 2023. https://doi.org/10.1007/978-981-99-1509-5_11.

(241) Li, P.; Pei, Y.; Li, J. A Comprehensive Survey on Design and Application of Autoencoder in Deep Learning. *Applied Soft Computing*. 2023. https://doi.org/10.1016/j.asoc.2023.110176.

(242) Baima, J.; Goryaeva, A. M.; Swinburne, T. D.; Maillet, J. B.; Nastar, M.; Marinica, M. C. Capabilities and Limits of Autoencoders for Extracting Collective Variables in Atomistic Materials Science. *Physical Chemistry Chemical Physics* **2022**, *24* (38). https://doi.org/10.1039/d2cp01917e.

(243) Kingma, D. P.; Welling, M. Auto-Encoding Variational Bayes. In *2nd International Conference on Learning Representations, ICLR 2014 - Conference Track Proceedings*; 2014.

(244) Sapkota, R.; Ahmed, D.; Karkee, M. Comparing YOLOv8 and Mask R-CNN for Instance Segmentation in Complex Orchard Environments. *Artificial Intelligence in Agriculture* **2024**, *13*. https://doi.org/10.1016/j.aiia.2024.07.001.

(245) He, K.; Gkioxari, G.; Dollár, P.; Girshick, R. Mask R-CNN. *IEEE Trans. Pattern Anal. Mach. Intell.* **2020**, *42* (2). https://doi.org/10.1109/TPAMI.2018.2844175.

(246) Ogunleye, A.; Wang, Q. G. XGBoost Model for Chronic Kidney Disease Diagnosis. *IEEE/ACM Trans. Comput. Biol. Bioinform.* **2020**, *17* (6). https://doi.org/10.1109/TCBB.2019.2911071.

(247) Ke, G.; Meng, Q.; Finley, T.; Wang, T.; Chen, W.; Ma, W.; Ye, Q.; Liu, T. Y. LightGBM: A Highly Efficient Gradient Boosting Decision Tree. In *Advances in Neural Information Processing Systems*; 2017; Vol. 2017-December.

(248) Hancock, J. T.; Khoshgoftaar, T. M. CatBoost for Big Data: An Interdisciplinary Review. *J. Big Data* **2020**, *7* (1). https://doi.org/10.1186/s40537-020-00369-8.

(249) Schulz, E.; Speekenbrink, M.; Krause, A. A Tutorial on Gaussian Process Regression: Modelling, Exploring, and Exploiting Functions. *J. Math. Psychol.* **2018**, *85*, 1–16. https://doi.org/10.1016/j.jmp.2018.03.001.

(250) Wang, X.; Jin, Y.; Schmitt, S.; Olhofer, M. Recent Advances in Bayesian Optimization. *ACM Comput. Surv.* **2023**, *55* (13s). https://doi.org/10.1145/3582078.

(251) Adams, R. P. A Tutorial on Bayesian Optimization for Machine Learning. -*Tutorial*- **2014**.

(252) Valleti, M.; Vasudevan, R. K.; Ziatdinov, M. A.; Kalinin, S. V. Bayesian Optimization in Continuous Spaces via Virtual Process Embeddings. *Digital Discovery* **2022**, *1* (6). https://doi.org/10.1039/d2dd00065b.

(253) Zhu, J.; Ma, C.; Zhang, Y.; Huang, H.; Kong, D.; Ni, W. Multi-Label Diagnosis of Arrhythmias Based on a Modified Two-Category Cross-Entropy Loss Function. *Electronics (Switzerland)* **2023**, *12* (24). https://doi.org/10.3390/electronics12244976.

(254) Connor, R.; Dearle, A.; Claydon, B.; Vadicamo, L. Correlations of Cross-Entropy Loss in Machine Learning. *Entropy* **2024**, *26* (6). https://doi.org/10.3390/e26060491.

(255) Wang, Q.; Ma, Y.; Zhao, K.; Tian, Y. A Comprehensive Survey of Loss Functions in Machine Learning. *Annals of Data Science* **2022**, *9* (2). https://doi.org/10.1007/s40745-020-00253-5.

(256) Raissi, M.; Perdikaris, P.; Karniadakis, G. E. Physics-Informed Neural Networks: A Deep Learning Framework for Solving Forward and Inverse Problems Involving Nonlinear Partial Differential Equations. *J. Comput. Phys.* **2019**, *378*. https://doi.org/10.1016/j.jcp.2018.10.045.

(257) Zhang, Z. Improved Adam Optimizer for Deep Neural Networks. In *2018 IEEE/ACM 26th International Symposium on Quality of Service, IWQoS 2018*; 2019. https://doi.org/10.1109/IWQoS.2018.8624183.

(258) Natekin, A.; Knoll, A. Gradient Boosting Machines, a Tutorial. *Front. Neurorobot.* **2013**, *7* (DEC). https://doi.org/10.3389/fnbot.2013.00021.

(259) Tian, Y.; Zhang, Y.; Zhang, H. Recent Advances in Stochastic Gradient Descent in Deep Learning. *Mathematics*. 2023. https://doi.org/10.3390/math11030682.

(260) Shao, Y.; Wang, J.; Sun, H.; Yu, H.; Xing, L.; Zhao, Q.; Zhang, L. An Improved BGE-Adam Optimization Algorithm Based on Entropy Weighting and Adaptive Gradient Strategy. *Symmetry (Basel).* **2024**, *16* (5). https://doi.org/10.3390/sym16050623.

(261) Friedman, J. H. Greedy Function Approximation: A Gradient Boosting Machine. *Ann. Stat.* **2001**, *29* (5). https://doi.org/10.1214/aos/1013203451.

(262) Kingma, D. P.; Ba, J. L. Adam: A Method for Stochastic Optimization. In *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*; 2015.

(263) Glorot, X.; Bengio, Y. Understanding the Difficulty of Training Deep Feedforward Neural Networks. In *Journal of Machine Learning Research*; 2010; Vol. 9.

(264) Peng, J.; Xiao, Y.; Imel, A.; Barth, B. A.; Cantillo, N. M.; Nelms, K. M. K.; Zawodzinski, T. A. Electrolyte Effects on the Electrochemical Performance of Microemulsions. *Electrochim. Acta* **2021**, *393*, 139048. https://doi.org/10.1016/J.ELECTACTA.2021.139048.

(265) Peng, J.; Cantillo, N. M.; Nelms, K. M.; Roberts, L. S.; Goenaga, G.; Imel, A.; Barth, B. A.; Dadmun, M.; Heroux, L.; Hayes, D. G.; Zawodzinski, T. Electron Transfer in Microemulsion-Based Electrolytes. *ACS Appl. Mater. Interfaces* **2020**, *12* (36), 40213–40219.

(266) Zou, H.; Wu, S.; Shen, J. Polymer/Silica Nanocomposites: Preparation, Characterization, Properties, and Applications. *Chem. Rev.* **2008**, *108* (9), 3893–3957. https://doi.org/10.1021/cr068035q.

(267) Eastoe, J.; Warne, B. Nanoparticle and Polymer Synthesis in Microemulsions. *Curr. Opin. Colloid Interface Sci.* **1996**, *1* (6), 800–805. https://doi.org/10.1016/s1359-0294(96)80084-7.

(268) Ulbrich, K.; Holá, K.; Šubr, V.; Bakandritsos, A.; Tuček, J.; Zbořil, R. Targeted Drug Delivery with Polymers and Magnetic Nanoparticles: Covalent and Noncovalent Approaches, Release Control, and Clinical Studies. *Chem. Rev.* **2016**, *116* (9), 5338–5431. https://doi.org/10.1021/acs.chemrev.5b00589.

(269) Bera, A.; Mandal, A. Microemulsions: A Novel Approach to Enhanced Oil Recovery: A Review. *J. Pet. Explor. Prod. Technol.* **2015**, *5* (3), 255–268. https://doi.org/10.1007/s13202-014-0139-5.

(270) Nazar, M. F.; Shah, S. S.; Khosa, M. A. Microemulsions in Enhanced Oil Recovery: A Review. *Pet. Sci. Technol.* **2011**, *29* (13), 1353–1365. https://doi.org/10.1080/10916460903502514.

(271) Liu, Y.; Chen, S.; Sheng, L.; Wang, M.; Liu, H. The Phase Behavior and Solubilization Ability of Nonionic Surfactant-Distillate Fraction of Crude Oil Microemulsion System. *Colloids Surf. A Physicochem. Eng. Asp.* **2020**, *603* (June), 125181. https://doi.org/10.1016/j.colsurfa.2020.125181.

(272) Winsor, P. A. Hydrotropy, Solubilisation and Related Emulsification Processes. Part I. **1947**, No. 1942.

(273) Tartaro, G.; Mateos, H.; Schirone, D.; Angelico, R.; Palazzo, G. Microemulsion Microstructure(s): A Tutorial Review. *Nanomaterials* **2020**, *10* (9), 1–40. https://doi.org/10.3390/nano10091657.

(274) Salager, J. L.; Morgan, J. C.; Schechter, R. S.; Wade, W. H.; Vasquez, E. Optimum Formulation of Surfactant/Water/Oil Systems for Minimum Interfacial Tension or Phase Behavior. *Society of Petroleum Engineers Journal* **1979**, *19* (02), 107–115. https://doi.org/10.2118/7054-PA.

(275) Bouton, F.; Durand, M.; Nardello-Rataj, V.; Serry, M.; Aubry, J. M. Classification of Terpene Oils Using the Fish Diagrams and the Equivalent Alkane Carbon (EACN) Scale. *Colloids Surf. A Physicochem. Eng. Asp.* **2009**, *338* (1–3), 142–147. https://doi.org/10.1016/j.colsurfa.2008.05.027.

(276) Nguyen, T. T.; Sabatini, D. A. Characterization and Emulsification Properties of Rhamnolipid and Sophorolipid Biosurfactants and Their Applications. *Int. J. Mol. Sci.* **2011**, *12* (2), 1232–1244. https://doi.org/10.3390/ijms12021232.

(277) Aubry, J. M.; Ontiveros, J. F.; Salager, J. L.; Nardello-Rataj, V. Use of the Normalized Hydrophilic-Lipophilic-Deviation (HLDN) Equation for Determining the Equivalent Alkane Carbon Number (EACN) of Oils and the Preferred Alkane Carbon Number (PACN) of Nonionic Surfactants by the Fish-Tail Method (FTM). *Adv. Colloid Interface Sci.* **2020**, *276*. https://doi.org/10.1016/j.cis.2019.102099.

(278) Wang, S.; Chen, C.; Yuan, N.; Ma, Y.; Ogbonnaya, O. I.; Shiau, B.; Harwell, J. H. Design of Extended Surfactant-Only EOR Formulations for an Ultrahigh Salinity Oil Field by Using Hydrophilic Lipophilic Deviation (HLD) Approach: From Laboratory Screening to Simulation. *Fuel* **2019**, *254*. https://doi.org/10.1016/j.fuel.2019.115698.

(279) Ghayour, A.; Acosta, E. J. Characterizing the Oil-like and Surfactant-like Behavior of Polar Oils. *Langmuir* **2019**. https://doi.org/10.1021/acs.langmuir.9b02732.

(280) Acosta, E. J.; Szekeres, E.; Sabatini, D. A.; Harwell, J. H. Net-Average Curvature Model for Solubilization and Supersolubilization in Surfactant Microemulsions. *Langmuir* **2003**, *38* (9), 186–195.

(281) Zarate-Muñoz, S.; Texeira De Vasconcelos, F.; Myint-Myat, K.; Minchom, J.; Acosta, E. A Simplified Methodology to Measure the Characteristic Curvature (Cc) of Alkyl Ethoxylate Nonionic Surfactants. *J. Surfactants Deterg.* **2016**, *19* (2), 249–263. https://doi.org/10.1007/s11743-016-1787-x.

(282) Ambikasaran, S.; Foreman-Mackey, D.; Greengard, L.; Hogg, D. W.; O'Neil, M. Fast Direct Methods for Gaussian Processes. *IEEE Trans. Pattern Anal. Mach. Intell.* **2016**, *38* (2). https://doi.org/10.1109/TPAMI.2015.2448083.

(283) Zhang, D.; Cai, Q.; Taiwo, O. O.; Yufit, V.; Brandon, N. P.; Gu, S. The Effect of Wetting Area in Carbon Paper Electrode on the Performance of Vanadium Redox

Flow Batteries: A Three-Dimensional Lattice Boltzmann Study. *Electrochim. Acta* **2018**, *283*. https://doi.org/10.1016/j.electacta.2018.07.027.

(284) Delforce, L.; Duprat, F.; Ploix, J. L.; Ontiveros, J. F.; Goussard, V.; Nardello-Rataj, V.; Aubry, J. M. Fast Prediction of the Equivalent Alkane Carbon Number Using Graph Machines and Neural Networks. *ACS Omega* **2022**, *7* (43), 38869–38881. https://doi.org/10.1021/acsomega.2c04592.

(285) Lukowicz, T.; Benazzouz, A.; Nardello-Rataj, V.; Aubry, J. M. Rationalization and Prediction of the Equivalent Alkane Carbon Number (EACN) of Polar Hydrocarbon Oils with COSMO-RS σ-Moments. *Langmuir* **2015**, *31* (41), 11220–11226. https://doi.org/10.1021/acs.langmuir.5b02545.

(286) Bouton, F.; Durand, M.; Nardello-Rataj, V.; Borosy, A. P.; Quellet, C.; Aubry, J. M. A QSPR Model for the Prediction of the "Fish-Tail" Temperature of CiE4/Water/Polar Hydrocarbon Oil Systems. *Langmuir* **2010**, *26* (11), 7962–7970. https://doi.org/10.1021/la904836m.

(287) Do, L. D.; Attaphong, C.; Scamehorn, J. F.; Sabatini, D. A. Detergency of Vegetable Oils and Semi-Solid Fats Using Microemulsion Mixtures of Anionic Extended Surfactants: The HLD Concept and Cold Water Applications. *J. Surfactants Deterg.* **2015**, *18* (3), 373–382. https://doi.org/10.1007/s11743-014-1659-1.

(288) Salager, J. L.; Antón, R.; Bullón, J.; Forgiarini, A.; Marquez, R. How to Use the Normalized Hydrophilic-Lipophilic Deviation (HLDN) Concept for the Formulation of Equilibrated and Emulsified Surfactant-Oil-Water Systems for Cosmetics and Pharmaceutical Products. *Cosmetics* **2020**, *7* (3), 1–50. https://doi.org/10.3390/cosmetics7030057.

(289) Tchakalova, V.; Fieber, W. Classification of Fragrances and Fragrance Mixtures Based on Interfacial Solubilization. *J. Surfactants Deterg.* **2012**, *15* (2), 167–177. https://doi.org/10.1007/s11743-011-1295-y.

(290) Mahboob, A.; Kalam, S.; Kamal, M. S.; Hussain, S. M. S.; Solling, T. EOR Perspective of Microemulsions: A Review. *J. Pet. Sci. Eng.* **2022**, *208* (PA), 109312. https://doi.org/10.1016/j.petrol.2021.109312.

(291) Artrith, N.; Butler, K. T.; Coudert, F.-X.; Han, S.; Isayev, O.; Jain, A.; Walsh, A. Best Practices in Machine Learning for Chemistry. *Nat. Chem.* **2021**, *13* (6), 505–508. https://doi.org/10.1038/s41557-021-00716-z.

(292) Oviedo, F.; Ferres, J. L.; Buonassisi, T.; Butler, K. T. Interpretable and Explainable Machine Learning for Materials Science and Chemistry. *Acc. Mater. Res.* **2022**, *3* (6), 597–607. https://doi.org/10.1021/accountsmr.1c00244.

(293) Qiao, C.; Yu, X.; Song, X.; Zhao, T.; Xu, X.; Zhao, S.; Gubbins, K. E. Enhancing Gas Solubility in Nanopores: A Combined Study Using Classical Density

Functional Theory and Machine Learning. *Langmuir* **2020**, *36* (29), 8527–8536. https://doi.org/10.1021/acs.langmuir.0c01160.

(294) Pan, C.; Mahmoudabadbozchelou, M.; Duan, X.; Benneyan, J. C.; Jamali, S.; Erb, R. M. Increasing Efficiency and Accuracy of Magnetic Interaction Calculations in Colloidal Simulation through Machine Learning. *J. Colloid Interface Sci.* **2022**, *611*, 29–38. https://doi.org/10.1016/J.JCIS.2021.11.195.

(295) Tung, C. H.; Chang, S. Y.; Chang, M. C.; Carrillo, J. M.; Sumpter, B. G.; Do, C.; Chen, W. R. Inferring Colloidal Interaction from Scattering by Machine Learning. *Carbon Trends* **2023**, *10*, 100252. https://doi.org/10.1016/J.CARTRE.2023.100252.

(296) Cencer, M. M.; Moore, J. S.; Assary, R. S. Machine Learning for Polymeric Materials: An Introduction. *Polym. Int.* **2022**, *71* (5), 537–542. https://doi.org/https://doi.org/10.1002/pi.6345.

(297) Abbott, S. *Surfactant Science: Principles and Practice*; DEStech Publications, 2019.

(298) Cayias, J. L.; Schechter, R. S.; Wade, W. H. Modeling Crude Oils for Low Interfacial Tension. *Society of Petroleum Engineers Journal* **1976**, *16* (06), 351–357. https://doi.org/10.2118/5813-PA.

(299) *Home page of cheminfo website.* http://www.cheminfo.org/ (accessed 2023-03-02).

(300) O'Boyle, N. M.; Banck, M.; James, C. A.; Morley, C.; Vandermeersch, T.; Hutchison, G. R. Open Babel: An Open Chemical Toolbox. *J. Cheminform.* **2011**, *3* (10). https://doi.org/10.1186/1758-2946-3-33.

(301) Halgren, T. A. MMFF VI. MMFF94s Option for Energy Minimization Studies. *J. Comput. Chem.* **1999**, *20* (7), 720–729. https://doi.org/10.1002/(sici)1096-987x(199905)20:7<720::aid-jcc7>3.0.co;2-x.

(302) Yoshikawa, N.; Hutchison, G. R. Fast, Efficient Fragment-Based Coordinate Generation for Open Babel. *J. Cheminform.* **2019**, *11* (1). https://doi.org/10.1186/s13321-019-0372-5.

(303) Brozos, C.; Bhattacharya, S.; Rittig, J. G.; Akanny, E.; Eiden, P.; Mitsos, A. Predicting the Equivalent Alkane Carbon Number of Oils Using Graph Neural Networks and Quantum Mechanical Descriptors. *Comput. Chem. Eng.* **2026**, *204*. https://doi.org/10.1016/j.compchemeng.2025.109425.

(304) Harris-Birtill, D.; Harris-Birtill, R. Understanding Computation Time: A Critical Discussion of Time as a Computational Performance Metric. In *Time in Variance*; BRILL, 2021; Vol. 17, pp 220–248. https://doi.org/10.1163/9789004470170_014.

(305) Sehgal, P.; Sharma, M.; Larsen, K. L.; Wimmer, R.; Otzen, D. E.; Doe, H. Influence of β-Cyclodextrin on the Mixed Micellization Process of Sodium Dodecyl Sulfate

and Sodium Lauroyl Sarcosine and Formation of Inclusion Complexes. *J. Dispers. Sci. Technol.* **2008**, *29* (1). https://doi.org/10.1080/01932690701688870.

(306) Ranajit Paul, M. K. M. R. N. P. B. Fluorescent Probe and NMR Studies of the Aggregation of Bile Salts in Aqueous Solution. *Chem. Phys. Lipids* **1979**, *25* (4), 345–356. https://doi.org/10.1016/0009-3084(79)90073-2.

(307) Kolesnikov, A.; Milyutina, A.; Desyatov, A.; Kolesnikov, V. Electroflotation Recovery of Highly Dispersed Carbon Materials from Aqueous Solutions of Electrolyte. *Sep. Purif. Technol.* **2019**, *209*. https://doi.org/10.1016/j.seppur.2018.07.014.

(308) Romero, C. M.; Escamilla, A. P.; Ribeiro, A. C. F.; Esteso, M. A. Effect of 1,2-Propanediol on the Critical Micelle Concentration of Decyltrimethylammonium Bromide at Temperatures from 293.15 K to 308.15 K. *Int. J. Mol. Sci.* **2022**, *23* (24). https://doi.org/10.3390/ijms232415884.

(309) Kroll, P.; Benke, J.; Enders, S.; Brandenbusch, C.; Sadowski, G. Influence of Temperature and Concentration on the Self-Assembly of Nonionic CiEj Surfactants: A Light Scattering Study. *ACS Omega* **2022**, *7* (8). https://doi.org/10.1021/acsomega.1c06766.

(310) Moura, S.; Furtado, A.; Pellegrino, O.; Lenard, E.; Malejczyk, E.; Sariyerli, G.; Akcadag, U.; Sousa, J. Surface Tension Measurements – A Comparative Study. *Acta IMEKO* **2023**, *12* (4). https://doi.org/10.21014/ACTAIMEKO.V12I4.1418.

(311) Vitková, Z.; Tárník, M.; Pavlovičová, J.; Vitko, A.; Oremusová, J.; Murgaš, J.; Babinec, A. From Static to Dynamic Modeling of Surfactants Micellization. *Tenside, Surfactants, Detergents* **2021**, *58* (1). https://doi.org/10.1515/tsd-2020-2296.

(312) Silva, G. M. C.; Pantano, D. A.; Loehlé, S.; Coutinho, J. A. P. The Challenges of Using COSMO-RS To Describe Polymer Solution Behavior. *Ind. Eng. Chem. Res.* **2023**, *62* (48). https://doi.org/10.1021/acs.iecr.3c03310.

(313) Klamt, A. The Beauty of Fluid Phase Thermodynamics from the Perspective of Quantum Chemistry. In *American Institute of Chemical Engineers 2008 Annual Meeting*; 2008.

(314) Schonlau, M.; Zou, R. Y. The Random Forest Algorithm for Statistical Learning. *Stata Journal* **2020**, *20* (1). https://doi.org/10.1177/1536867X20909688.

(315) Anggoro, D. A.; Afdallah, N. A. Grid Search CV Implementation in Random Forest Algorithm to Improve Accuracy of Breast Cancer Data. *Int. J. Adv. Sci. Eng. Inf. Technol.* **2022**, *12* (2). https://doi.org/10.18517/ijaseit.12.2.15487.

(316) Harris-Birtill, D.; Harris-Birtill, R. Understanding Computation Time: A Critical Discussion of Time as a Computational Performance Metric. In *Study of Time*; 2021; Vol. 17. https://doi.org/10.1163/9789004470170_014.

(317) Chen, L. J.; Lin, S. Y.; Huang, C. C. Effect of Hydrophobic Chain Length of Surfactants on Enthalpy-Entropy Compensation of Micellization. *Journal of Physical Chemistry B* **1998**, *102* (22). https://doi.org/10.1021/jp9804345.

(318) Oremusová, J.; Vitková, Z.; Vitko, A.; Tárník, M.; Miklovičová, E.; Ivánková, O.; Murgaš, J.; Krchňák, D. Effect of Molecular Composition of Head Group and Temperature on Micellar Properties of Ionic Surfactants with C12 Alkyl Chain. *Molecules* **2019**, *24* (3). https://doi.org/10.3390/molecules24030651.

(319) Yang, W.; Cao, Y.; Wang, Y.; Ju, H.; Jiang, Y.; Geng, T. Effects of Unsaturated Double Bonds on Adsorption and Aggregation Behaviors of Amide-Based Cationic Gemini Surfactants. *Colloids Surf. A Physicochem. Eng. Asp.* **2021**, *611*. https://doi.org/10.1016/j.colsurfa.2020.125778.

(320) Sprague, E. D.; Duecker, D. C.; Larrabee, C. E. The Effect of a Terminal Double Bond on the Micellization of a Simple Ionic Surfactant. *J. Colloid Interface Sci.* **1983**, *92* (2). https://doi.org/10.1016/0021-9797(83)90163-7.

(321) Mozrzymas, A. Designing the Structure of Cationic Star-Shaped Trimeric Surfactants Most Active in Micelle Formation Using Molecular Connectivity Indices. *Sci. Rep.* **2024**, *14* (1). https://doi.org/10.1038/s41598-024-58854-6.

(322) Gomes, A. M. M.; Granjo, J. F. O.; Costa, P. J. Conformer Generation Workflows for COSMO-RS Calculations: Are They All the Same? November 25, 2025. https://doi.org/10.26434/chemrxiv-2025-2ts7t.

(323) Dolfini, A.; Angelini, I.; Artioli, G. Copper to Tuscany – Coals to Newcastle? The Dynamics of Metalwork Exchange in Early Italy. *PLoS One* **2020**, *15* (1). https://doi.org/10.1371/journal.pone.0227259.

(324) Danjo, K.; Kamiya, K.; Otsuka, A. Effect of Temperature on the Sticking of Low Melting Point Materials. *Chem. Pharm. Bull. (Tokyo).* **1993**, *41* (8). https://doi.org/10.1248/cpb.41.1423.

(325) Liu, H.; Zhou, L. Progress in Friction Stir Welding of High Melting Point Materials. *Hanjie Xuebao/Transactions of the China Welding Institution*. 2007.

(326) Hidayanti, F.; Yulianto, T.; Wismogroho, A. S. DESIGN OF DIFFERENTIAL THERMAL ANALYSIS TOOL AIDS FOR MELTING POINT MATERIAL ANALYSIS. *International Journal of Applied Science and Engineering Review* **2023**, *04* (06). https://doi.org/10.52267/ijaser.2023.4601.

(327) Li, Q.; He, G.; Ding, Y. Applications of Low-Melting-Point Metals in Rechargeable Metal Batteries. *Chemistry - A European Journal*. 2021. https://doi.org/10.1002/chem.202003921.

(328) Hoshi, A.; Mills, D. R.; Bittar, A.; Saitoh, T. S. Screening of High Melting Point Phase Change Materials (PCM) in Solar Thermal Concentrating Technology Based on CLFR. *Solar Energy* **2005**, *79* (3). https://doi.org/10.1016/j.solener.2004.04.023.

(329) Habibi-Yangjeh, A.; Pourbasheer, E.; Danandeh-Jenagharad, M. Prediction of Melting Point for Drug-like Compounds Using Principal Component-Genetic Algorithm-Artificial Neural Network. *Bull. Korean Chem. Soc.* **2008**, *29* (4). https://doi.org/10.5012/bkcs.2008.29.4.833.

(330) Huang, D.; Xie, Z.; Rao, Q.; Liamas, E.; Pan, P.; Guan, S.; Zhang, Z. J.; Lu, M.; Li, Q. Hot Melt Extrusion of Heat-Sensitive and High Melting Point Drug: Inhibit the Recrystallization of the Prepared Amorphous Drug during Extrusion to Improve the Bioavailability. *Int. J. Pharm.* **2019**, *565*. https://doi.org/10.1016/j.ijpharm.2019.04.064.

(331) Park, J. H.; Kwon, D. Y.; Heo, J. Y.; Park, S. H.; Park, J. Y.; Lee, B.; Kim, J. H.; Kim, M. S. Effect of Drug Carrier Melting Points on Drug Release of Dexamethasone-Loaded Microspheres. *Tissue Eng. Regen. Med.* **2017**, *14* (6). https://doi.org/10.1007/s13770-017-0077-7.

(332) Bergström, C. A. S.; Norinder, U.; Luthman, K.; Artursson, P. Molecular Descriptors Influencing Melting Point and Their Role in Classification of Solid Drugs. *J. Chem. Inf. Comput. Sci.* **2003**, *43* (4). https://doi.org/10.1021/ci020280x.

(333) Bradley, J.-C.; Lang, A.; Williams, A. Jean-Claude Bradley Double plus Good (Highly Curated and Validated) Melting Point Dataset. *Figshare* **2014**, *10*.

(334) Tetko, I. V.; M. Lowe, D.; Williams, A. J. The Development of Models to Predict Melting and Pyrolysis Point Data Associated with Several Hundred Thousand Compounds Mined from PATENTS. *J. Cheminform.* **2016**, *8* (1). https://doi.org/10.1186/s13321-016-0113-y.

(335) P, D. CRC Handbook of Chemistry and Physics. *J. Mol. Struct.* **1992**, *268* (1–3). https://doi.org/10.1016/0022-2860(92)85083-s.

(336) Schrier, J. Can One Hear the Shape of a Molecule (from Its Coulomb Matrix Eigenvalues)? *J. Chem. Inf. Model.* **2020**, *60* (8). https://doi.org/10.1021/acs.jcim.0c00631.

(337) Quinlan, J. R. Induction of Decision Trees. *Mach. Learn.* **1986**, *1* (1). https://doi.org/10.1023/A:1022643204877.

(338) Sanchez-Lengeling, B.; Roch, L. M.; Perea, J. D.; Langner, S.; Brabec, C. J.; Aspuru-Guzik, A. A Bayesian Approach to Predict Solubility Parameters. *Adv. Theory Simul.* **2019**, *2* (1). https://doi.org/10.1002/adts.201800069.

(339) Bhola, R.; Vaghani, H.; Bhatt, K.; Ghumara, R. Drug Solubility and Dissolution Thermodynamic Approach in Various Solvents at Different Temperatures: A Review. *Biointerface Research in Applied Chemistry*. 2022. https://doi.org/10.33263/BRIAC124.43744383.

(340) Pathakoti, K.; Manubolu, M.; Hwang, H. M. Nanostructures: Current Uses and Future Applications in Food Science. *Journal of Food and Drug Analysis*. 2017. https://doi.org/10.1016/j.jfda.2017.02.004.

(341) Liang, Y.; Job, H.; Feng, R.; Parks, F.; Hollas, A.; Zhang, X.; Bowden, M.; Noh, J.; Murugesan, V.; Wang, W. High-Throughput Solubility Determination for Data-Driven Materials Design and Discovery in Redox Flow Battery Research. *Cell Rep. Phys. Sci.* **2023**, *4* (10). https://doi.org/10.1016/j.xcrp.2023.101633.

(342) Khadka, P.; Ro, J.; Kim, H.; Kim, I.; Kim, J. T.; Kim, H.; Cho, J. M.; Yun, G.; Lee, J. Pharmaceutical Particle Technologies: An Approach to Improve Drug Solubility, Dissolution and Bioavailability. *Asian Journal of Pharmaceutical Sciences*. 2014. https://doi.org/10.1016/j.ajps.2014.05.005.

(343) Ma, Y.; Gao, Z.; Shi, P.; Chen, M.; Wu, S.; Yang, C.; Wang, J.; Cheng, J.; Gong, J. Machine Learning-Based Solubility Prediction and Methodology Evaluation of Active Pharmaceutical Ingredients in Industrial Crystallization. *Front. Chem. Sci. Eng.* **2022**, *16* (4), 523–535. https://doi.org/10.1007/s11705-021-2083-5.

(344) Klamt, A.; Eckert, F.; Hornig, M.; Beck, M. E.; Brger, T. Prediction of Aqueous Solubility of Drugs and Pesticides with COSMO-RS. *J. Comput. Chem.* **2002**, *23* (2). https://doi.org/10.1002/jcc.1168.

(345) Tian, Y.; Wang, X.; Liu, Y.; Hu, W. Prediction of Nitrogen Solubility in Ionic Liquids by Machine Learning Methods Based on COSMO-Derived Descriptors. *Chem. Eng. Sci.* **2024**, *284*. https://doi.org/10.1016/j.ces.2023.119482.

(346) Wang, J.; Song, Z.; Chen, L.; Xu, T.; Deng, L.; Qi, Z. Prediction of CO2 Solubility in Deep Eutectic Solvents Using Random Forest Model Based on COSMO-RS-Derived Descriptors. *Green Chemical Engineering* **2021**, *2* (4). https://doi.org/10.1016/j.gce.2021.08.002.

(347) Schneider, G. Prediction of Drug-Like Properties. In *Adaptive Systems in Drug Design*; 2020. https://doi.org/10.1201/9781498713702-10.

(348) Vassileiou, A. D.; Robertson, M. N.; Wareham, B. G.; Soundaranathan, M.; Ottoboni, S.; Florence, A. J.; Hartwig, T.; Johnston, B. F. A Unified ML Framework for Solubility Prediction across Organic Solvents. *Digital Discovery* **2023**, *2* (2). https://doi.org/10.1039/d2dd00024e.

(349) Khurana, S.; Rawi, R.; Kunji, K.; Chuang, G. Y.; Bensmail, H.; Mall, R. DeepSol: A Deep Learning Framework for Sequence-Based Protein Solubility Prediction. *Bioinformatics* **2018**, *34* (15). https://doi.org/10.1093/bioinformatics/bty166.

(350) Ye, Z.; Ouyang, D. Prediction of Small-Molecule Compound Solubility in Organic Solvents by Machine Learning Algorithms. *J. Cheminform.* **2021**, *13* (1). https://doi.org/10.1186/s13321-021-00575-3.

(351) Tayyebi, A.; Alshami, A. S.; Rabiei, Z.; Yu, X.; Ismail, N.; Talukder, M. J.; Power, J. Prediction of Organic Compound Aqueous Solubility Using Machine Learning: A Comparison Study of Descriptor-Based and Fingerprints-Based Models. *J. Cheminform.* **2023**, *15* (1). https://doi.org/10.1186/s13321-023-00752-6.

(352) Perryman, A. L.; Inoyama, D.; Patel, J. S.; Ekins, S.; Freundlich, J. S. Pruned Machine Learning Models to Predict Aqueous Solubility. *ACS Omega* **2020**, *5* (27). https://doi.org/10.1021/acsomega.0c01251.

(353) Van Erven, T.; Harrëmos, P. Rényi Divergence and Kullback-Leibler Divergence. *IEEE Trans. Inf. Theory* **2014**, *60* (7). https://doi.org/10.1109/TIT.2014.2320500.

(354) Nazábal, A.; Olmos, P. M.; Ghahramani, Z.; Valera, I. Handling Incomplete Heterogeneous Data Using VAEs. *Pattern Recognit.* **2020**, *107*. https://doi.org/10.1016/j.patcog.2020.107501.

(355) Cortes, C.; Mohri, M.; Rostamizadeh, A. L2 Regularization for Learning Kernels. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence, UAI 2009*; AUAI Press, 2009; pp 109–116.

(356) Ng, A. Y. Feature Selection, L1 vs. L2 Regularization, and Rotational Invariance. In *Proceedings, Twenty-First International Conference on Machine Learning, ICML 2004*; 2004; pp 615–622. https://doi.org/10.1145/1015330.1015435.

(357) Wong, T. T.; Yeh, P. Y. Reliable Accuracy Estimates from K-Fold Cross Validation. *IEEE Trans. Knowl. Data Eng.* **2020**, *32* (8). https://doi.org/10.1109/TKDE.2019.2912815.

(358) Lipinski, C. A.; Lombardo, F.; Dominy, B. W.; Feeney, P. J. Experimental and Computational Approaches to Estimate Solubility and Permeability in Drug Discovery and Development Settings. *Advanced Drug Delivery Reviews*. 2012. https://doi.org/10.1016/j.addr.2012.09.019.

(359) Kappal, S. Data Normalization Using Median & Median Absolute Deviation (MMAD) Based Z-Score for Robust Predictions vs. Min-Max Normalization. *London Journal of Research in Science: Natural and Formal* **2019**, *19* (4).

(360) Bertz, S. H. The First General Index of Molecular Complexity. *J. Am. Chem. Soc.* **1981**, *103* (12). https://doi.org/10.1021/ja00402a071.

(361) Huang, A. A.; Huang, S. Y. Increasing Transparency in Machine Learning through Bootstrap Simulation and Shapely Additive Explanations. *PLoS One* **2023**, *18* (2 February). https://doi.org/10.1371/journal.pone.0281922.

(362) Boobier, S.; Hose, D. R. J.; Blacker, A. J.; Nguyen, B. N. Machine Learning with Physicochemical Relationships: Solubility Prediction in Organic Solvents and Water. *Nat. Commun.* **2020**, *11* (1). https://doi.org/10.1038/s41467-020-19594-z.

(363) Yohannes, A. G.; Fink, K.; Kondov, I. Pt Nanoparticles under Oxidizing Conditions - Implications of Particle Size, Adsorption Sites and Oxygen Coverage on Stability. *Nanoscale Adv.* **2022**. https://doi.org/10.1039/d2na00490a.

(364) Lauf, P.; Morales, A. L.; Ehelebe, K.; Hutzler, A.; Mayrhofer, K. J. J.; Lloret, V.; Cherevko, S. Benchmarking Performance and Degradation of Pt Supported on Different Carbons for PEMFC Applications Using Gas Diffusion Electrode Half-Cell. *ACS Appl. Energy Mater.* **2025**, *8* (12). https://doi.org/10.1021/acsaem.5c00184.

(365) Yu, K.; Li, C.; Xie, J.; Ferreira, P. J. Understanding the Degradation Mechanisms of Pt Electrocatalysts in PEMFCs by Combining 2D and 3D Identical Location TEM. *Nano Lett.* **2023**, *23* (5). https://doi.org/10.1021/acs.nanolett.2c04764.

(366) Kregar, A.; Kravos, A.; Katrašnik, T. Methodology for Evaluation of Contributions of Ostwald Ripening and Particle Agglomeration to Growth of Catalyst Particles in PEM Fuel Cells. *Fuel Cells* **2020**, *20* (4). https://doi.org/10.1002/fuce.201900208.

(367) Eliasson, H.; Erni, R. Improving Nanoparticle Size Estimation from Scanning Transmission Electron Micrographs with a Multislice Surrogate Model. *Nano Lett.* **2025**, *25* (6). https://doi.org/10.1021/acs.nanolett.4c06025.

(368) Parthasarathy, P.; Virkar, A. V. Electrochemical Ostwald Ripening of Pt and Ag Catalysts Supported on Carbon. *J. Power Sources* **2013**, *234*. https://doi.org/10.1016/j.jpowsour.2013.01.115.

(369) Pelz, P. M.; Rakowski, A.; Rangel Dacosta, L.; Savitzky, B. H.; Scott, M. C.; Ophus, C. A Fast Algorithm for Scanning Transmission Electron Microscopy Imaging and 4D-STEM Diffraction Simulations. *Microscopy and Microanalysis* **2021**, *27* (4). https://doi.org/10.1017/S1431927621012083.

(370) Ooe, K.; Sadri, A.; Findlay, S. D. Optimizing Structure Factors From Four-Dimensional Scanning Transmission Electron Microscopy via the Scattering Matrix. *Microscopy and Microanalysis* **2025**, *31* (6). https://doi.org/10.1093/mam/ozaf111.

(371) Laramy, C. R.; Brown, K. A.; O'Brien, M. N.; Mirkin, C. A. High-Throughput, Algorithmic Determination of Nanoparticle Structure from Electron Microscopy Images. *ACS Nano* **2015**, *9* (12). https://doi.org/10.1021/acsnano.5b05968.

(372) Bell, C. G.; Treder, K. P.; Kim, J. S.; Schuster, M. E.; Kirkland, A. I.; Slater, T. J. A. Trainable Segmentation for Transmission Electron Microscope Images of

Inorganic Nanoparticles. *J. Microsc.* **2022**, *288* (3). https://doi.org/10.1111/jmi.13110.

(373) Hayden, L.; Strausborger, S.; Lewin-Smith, M. Automated Particle Analysis Using Field-Emission Scanning Electron Microscopy (FE-SEM) and Energy Dispersive X-Ray Spectroscopy (EDS) to Characterize Inhaled Particulate Matter (PM) in Biopsied Lung Tissue. *Microscopy and Microanalysis* **2023**, *29* (1). https://doi.org/10.1093/micmic/ozac015.

(374) Lu, X. (Lucy); He, K. Machine Learning Enhanced Image Segmentation for High-Fidelity STEM Data Analysis. *Microscopy and Microanalysis* **2023**, *29* (Supplement_1). https://doi.org/10.1093/micmic/ozad067.978.

(375) Ferji, K. DetectNano: Deep Learning Detection in TEM Images for High-Throughput Nanostructure Characterization. *Nanoscale* **2025**, *17* (32). https://doi.org/10.1039/d5nr02446c.

(376) Eliasson, H.; Erni, R. Localization and Segmentation of Atomic Columns in Supported Nanoparticles for Fast Scanning Transmission Electron Microscopy. *NPJ Comput. Mater.* **2024**, *10* (1). https://doi.org/10.1038/s41524-024-01360-0.

(377) Walker, H. A.; Pfeifer, T. W.; Zeiger, P. M.; Hachtel, J. A.; Pantelides, S. T.; Hoglund, E. R. PySlice: Routine Vibrational Electron Energy Loss Spectroscopy Prediction with Universal Interatomic Potentials. **2026**.

(378) Kirkland, E. J. *Advanced Computing in Electron Microscopy*; 2020. https://doi.org/10.1007/978-3-030-33260-0.

(379) Al-Ameen, Z.; Al-Ameen, S.; Al-Othman, A. Improving the Sharpness of Digital Images Using a Modified Laplacian Sharpening Technique. *IPTEK The Journal for Technology and Science* **2019**, *29* (2). https://doi.org/10.12962/j20882033.v29i2.3356.

(380) Vijaya Madhavi, V.; Kumari, P. L. S. A Qualitative Approach for Enhancing Fundus Images with Novel CLAHE Methods. *Engineering, Technology and Applied Science Research* **2025**, *15* (1). https://doi.org/10.48084/etasr.9525.

APPENDIX

Table A.1. List and descriptions of RDKit descriptors

Name	Descriptor
exactmw	The exact molecular weight of a molecule in Daltons
amw	The average molecular weight of a molecule in Daltons
lipinskiHBA	The number of hydrogen bond acceptors in a molecule, as defined by Lipinski's rule of five
lipinskiHBD	The number of hydrogen bond donors in a molecule, as defined by Lipinski's rule of five
NumRotatableBonds	The number of rotatable bonds in a molecule
NumHBD	The number of hydrogen bond donors in a molecule
NumHBA	The number of hydrogen bond acceptors in a molecule
NumHeavyAtoms	The number of non-hydrogen atoms in a molecule
NumAtoms	The number of atoms in a molecule
NumHeteroatoms	The number of heteroatoms (atoms other than carbon and hydrogen) in a molecule
NumAmideBonds	The number of amide bonds in a molecule
FractionCSP3	The fraction of carbons in a molecule that are sp3 hybridized
NumRings	The number of rings in a molecule
NumAromaticRings	The number of aromatic rings in a molecule
NumAliphaticRings	The number of aliphatic (non-aromatic) rings in a molecule

Table A.1. continued. List and descriptions of RDKit descriptors

NumSaturatedRings	The number of saturated rings in a molecule
NumHeterocycles	The number of heterocyclic rings in a molecule
NumAromaticHeterocycles	The number of aromatic heterocyclic rings in a molecule
NumSaturatedHeterocycles	The number of saturated heterocyclic rings in a molecule
NumAliphaticHeterocycles	The number of aliphatic heterocyclic rings in a molecule
NumSpiroAtoms	The number of spiro atoms in a molecule
NumBridgeheadAtoms	The number of bridgehead atoms in a molecule
NumAtomStereoCenters	The number of atoms in a molecule that are stereo centers
NumUnspecifiedAtomStereoCenters	The number of atoms in a molecule that are stereo centers but have unspecified stereochemistry
LabuteASA	The Labute's Approximate Surface Area of a molecule
TPSA	The Topological Polar Surface Area of a molecule
CrippenClogP	The Crippen's calculated octanol-water partition coefficient of a molecule
CrippenMR	The Crippen's calculated molar refractivity of a molecule
Chi0	The zeroth order valence connectivity index of a molecule
Chi0n	The first order valence connectivity index of a molecule

Table A.1. continued. List and descriptions of RDKit descriptors

Chi0v	The second order valence connectivity index of a molecule
Chi1	The third order valence connectivity index of a molecule
Chi1n	The fourth order valence connectivity index of a molecule
Chi1v	The zeroth order valence connectivity index including only non-hydrogen atoms of a molecule
Chi2n	The first order valence connectivity index including only non-hydrogen atoms of a molecule
Chi2v	The second order valence connectivity index including only non-hydrogen atoms of a molecule
Chi3n	The third order valence connectivity index including only non-hydrogen atoms of a molecule
Chi3v	The fourth order valence connectivity index including only non-hydrogen atoms of a molecule
HallKierAlpha	The Hall-Kier alpha shape index of a molecule
Kappa1	The first order molecular shape index of a molecule
Kappa2	The second order molecular shape index of a molecule
Kappa3	The third order molecular shape index of a molecule
Phi	The energy of the highest occupied molecular orbital (HOMO) minus the energy of the lowest unoccupied molecular orbital (LUMO) of a molecule
BalabanJ	The Balaban's J index of a molecule

Table A.1. continued. List and descriptions of RDKit descriptors

BertzCT	The Bertz's complexity index of a molecule
EState_VSA1	The estate value sum over van der Waals surface area of a molecule for atoms with estate values between 0.00 and 0.27
EState_VSA10	The estate value sum over van der Waals surface area of a molecule for atoms with estate values between 3.00 and 3.27
EState_VSA11	The estate value sum over van der Waals surface area of a molecule for atoms with estate values between 3.27 and 3.94
EState_VSA2	The estate value sum over van der Waals surface area of a molecule for atoms with estate values between 0.27 and 0.81
EState_VSA3	The estate value sum over van der Waals surface area of a molecule for atoms with estate values between 0.81 and 1.23
EState_VSA4	The estate value sum over van der Waals surface area of a molecule for atoms with estate values between 1.23 and 1.44
EState_VSA5	The estate value sum over van der Waals surface area of a molecule for atoms with estate values between 1.44 and 1.65
EState_VSA6	The estate value sum over van der Waals surface area of a molecule for atoms with estate values between 1.65 and 2.00

Table A.1. continued. List and descriptions of RDKit descriptors

EState_VSA7	The estate value sum over van der Waals surface area of a molecule for atoms with estate values between 2.00 and 2.47
EState_VSA8	The estate value sum over van der Waals surface area of a molecule for atoms with estate values between 2.47 and 3.00
EState_VSA9	The estate value sum over van der Waals surface area of a molecule for atoms with estate values between 3.00 and 3.27
FpDensityMorgan1	The Morgan fingerprint density of a molecule with radius 1
FpDensityMorgan2	The Morgan fingerprint density of a molecule with radius 2
FpDensityMorgan3	The Morgan fingerprint density of a molecule with radius 3
FractionCSP3	The fraction of carbons in a molecule that are sp3 hybridized
HeavyAtomCount	The number of heavy atoms (non-hydrogen atoms) in a molecule
HeavyAtomMolWt	The molecular weight of a molecule excluding hydrogen
Ipc	The Balaban's J index of a molecule
MaxAbsEStateIndex	The maximum absolute value of the E-state indices of atoms in a molecule
MaxAbsPartialCharge	The maximum absolute value of the partial charges of atoms in a molecule

Table A.1. continued. List and descriptions of RDKit descriptors

MaxEStateIndex	The maximum E-state index of atoms in a molecule
MaxPartialCharge	The maximum partial charge of atoms in a molecule
MinAbsEStateIndex	The minimum absolute value of the E-state indices of atoms in a molecule
MinAbsPartialCharge	The minimum absolute value of the partial charges of atoms in a molecule
MinEStateIndex	The minimum E-state index of atoms in a molecule
MinPartialCharge	The minimum partial charge of atoms in a molecule
MolLogP	The octanol-water partition coefficient of a molecule
MolMR	The molecular refractivity of a molecule
MolWt	The molecular weight of a molecule in Daltons
NHOHCount	The number of hydroxyl and amine groups in a molecule
NOCount	The number of nitro and nitroso groups in a molecule
NumAliphaticCarbocycles	The number of aliphatic (non-aromatic) carbocyclic rings in a molecule
NumAliphaticHeterocycles	The number of aliphatic (non-aromatic) heterocyclic rings in a molecule
NumAliphaticRings	The number of aliphatic (non-aromatic) rings in a molecule
NumAromaticCarbocycles	The number of aromatic carbocyclic rings in a molecule

Table A.1. continued. List and descriptions of RDKit descriptors

NumAromaticHeterocycles	The number of aromatic heterocyclic rings in a molecule
NumAromaticRings	The number of aromatic rings in a molecule
NumHAcceptors	The number of hydrogen bond acceptors in a molecule
NumHDonors	The number of hydrogen bond donors in a molecule
NumHeteroatoms	The number of heteroatoms (atoms other than carbon and hydrogen) in a molecule
NumRadicalElectrons	The number of radical electrons in a molecule
NumRotatableBonds	The number of rotatable bonds in a molecule
NumSaturatedCarbocycles	The number of saturated carbocyclic rings in a molecule
NumSaturatedHeterocycles	The number of saturated heterocyclic rings in a molecule
NumSaturatedRings	The number of saturated rings in a molecule
NumValenceElectrons	The number of valence electrons in a molecule
PEOE_VSA1	Partial positive surface area of the molecule with a probe radius of 1.4 Angstroms
PEOE_VSA10	Partial positive surface area of the molecule with a probe radius of 10.0 Angstroms
PEOE_VSA11	Partial positive surface area of the molecule with a probe radius of 11.0 Angstroms
PEOE_VSA12	Partial positive surface area of the molecule with a probe radius of 12.0 Angstroms

Table A.1. continued. List and descriptions of RDKit descriptors

PEOE_VSA13	Partial positive surface area of the molecule with a probe radius of 13.0 Angstroms
PEOE_VSA14	Partial positive surface area of the molecule with a probe radius of 14.0 Angstroms
PEOE_VSA2	Partial positive surface area of the molecule with a probe radius of 2.0 Angstroms
PEOE_VSA3	Partial positive surface area of the molecule with a probe radius of 3.0 Angstroms
PEOE_VSA4	Partial positive surface area of the molecule with a probe radius of 4.0 Angstroms
PEOE_VSA5	Partial positive surface area of the molecule with a probe radius of 5.0 Angstroms
PEOE_VSA6	Partial positive surface area of the molecule with a probe radius of 6.0 Angstroms
PEOE_VSA7	Partial positive surface area of the molecule with a probe radius of 7.0 Angstroms
PEOE_VSA8	Partial positive surface area of the molecule with a probe radius of 8.0 Angstroms
PEOE_VSA9	Partial positive surface area of the molecule with a probe radius of 9.0 Angstroms
RingCount	Number of rings in the molecule
SMR_VSA1	Scaled mean polarizability of the molecule with a probe radius of 1.4 Angstroms
SMR_VSA10	Scaled mean polarizability of the molecule with a probe radius of 10.0 Angstroms

Table A.1. continued. List and descriptions of RDKit descriptors

SMR_VSA2	Scaled mean polarizability of the molecule with a probe radius of 2.0 Angstroms
SMR_VSA3	Scaled mean polarizability of the molecule with a probe radius of 3.0 Angstroms
SMR_VSA4	Scaled mean polarizability of the molecule with a probe radius of 4.0 Angstroms
SMR_VSA5	Scaled mean polarizability of the molecule with a probe radius of 5.0 Angstroms
SMR_VSA6	Scaled mean polarizability of the molecule with a probe radius of 6.0 Angstroms
SMR_VSA7	Scaled mean polarizability of the molecule with a probe radius of 7.0 Angstroms
SMR_VSA8	Scaled mean polarizability of the molecule with a probe radius of 8.0 Angstroms
SMR_VSA9	Scaled mean polarizability of the molecule with a probe radius of 9.0 Angstroms
SlogP_VSA1	Partial negative surface area of the molecule with a probe radius of 1.4 Angstroms
SlogP_VSA10	Partial negative surface area of the molecule with a probe radius of 10.0 Angstroms
SlogP_VSA11	Partial negative surface area of the molecule with a probe radius of 11.0 Angstroms
SlogP_VSA12	Partial negative surface area of the molecule with a probe radius of 12.0 Angstroms

Table A.1. continued. List and descriptions of RDKit descriptors

SlogP_VSA2	Partial negative surface area of the molecule with a probe radius of 2.0 Angstroms
SlogP_VSA3	Partial negative surface area of the molecule with a probe radius of 3.0 Angstroms
SlogP_VSA4	Partial negative surface area of the molecule with a probe radius of 4.0 Angstroms
SlogP_VSA5	Partial negative surface area of the molecule with a probe radius of 5.0 Angstroms
SlogP_VSA6	Partial negative surface area of the molecule with a probe radius of 6.0 Angstroms
SlogP_VSA7	Partial negative surface area of the molecule with a probe radius of 7.0 Angstroms
SlogP_VSA8	Partial negative surface area of the molecule with a probe radius of 8.0 Angstroms
SlogP_VSA9	Partial negative surface area of the molecule with a probe radius of 9.0 Angstroms
VSA_EState1	Calculates the distribution of the electrostatic potential in a molecule, which is useful for predicting the biological activity of a compound with x of ($-\infty < x < 4.78$)
VSA_EState10	Calculates the distribution of the electrostatic potential in a molecule, which is useful for predicting the biological activity of a compound with x of ($11.00 \leq x < \infty$)

Table A.1. continued. List and descriptions of RDKit descriptors

VSA_EState2	Calculates the distribution of the electrostatic potential in a molecule, which is useful for predicting the biological activity of a compound with x of ($4.78 \leq x < 5.00$)
VSA_EState3	Calculates the distribution of the electrostatic potential in a molecule, which is useful for predicting the biological activity of a compound with x of ($5.00 \leq x < 5.41$)
VSA_EState4	Calculates the distribution of the electrostatic potential in a molecule, which is useful for predicting the biological activity of a compound with x of ($5.41 \leq x < 5.74$)
VSA_EState5	Calculates the distribution of the electrostatic potential in a molecule, which is useful for predicting the biological activity of a compound with x of ($5.74 \leq x < 6.00$)
VSA_EState6	Calculates the distribution of the electrostatic potential in a molecule, which is useful for predicting the biological activity of a compound with x of ($6.00 \leq x < 6.07$)
VSA_EState7	Calculates the distribution of the electrostatic potential in a molecule, which is useful for predicting the biological activity of a compound with x of ($6.45 \leq x < 7.00$)
VSA_EState8	Calculates the distribution of the electrostatic potential in a molecule, which is useful for predicting the biological activity of a compound with x of ($6.45 \leq x < 7.00$)

Table A.1. continued. List and descriptions of RDKit descriptors

VSA_EState9	Calculates the distribution of the electrostatic potential in a molecule, which is useful for predicting the biological activity of a compound with x of ($7.00 \leq x < 11.00$)
fr_Al_COO	Number of carboxylic acid groups attached to an aromatic ring
fr_Al_OH	Number of hydroxyl groups attached to an aromatic ring
fr_Al_OH_noTert	Number of hydroxyl groups attached to an aromatic ring, excluding those attached to a tertiary carbon
fr_ArN	Number of nitrogen atoms in an aromatic ring
fr_Ar_COO	Number of carboxylic acid groups attached to an aromatic ring
fr_Ar_N	Number of nitrogen atoms in an aromatic ring
fr_Ar_NH	Number of NH groups attached to an aromatic ring
fr_Ar_OH	Number of hydroxyl groups attached to an aromatic ring
fr_COO	Number of carboxylic acid groups
fr_COO2	Number of dicarboxylic acid groups
fr_C_O	Number of carbonyl groups (C=O)
fr_C_O_noCOO	Number of carbonyl groups (C=O) not attached to a carboxylic acid group
fr_C_S	Number of thiocarbonyl groups (C=S)

Table A.1. continued. List and descriptions of RDKit descriptors

fr_HOCCN	Number of amide groups (C(=O)N)
fr_Imine	Number of imine groups (C=N)
fr_NH0	Number of primary amine groups (NH2)
fr_NH1	Number of secondary amine groups (NH)
fr_NH2	Number of tertiary amine groups (N)
fr_N_O	Number of N-O groups
fr_Ndealkylation1	Number of primary amines that undergo dealkylation
fr_Ndealkylation2	Number of secondary amines that undergo dealkylation
fr_Nhpyrrole	Number of pyrrole nitrogen atoms
fr_SH	Number of thiols
fr_aldehyde	Number of aldehydes
fr_alkyl_carbamate	Number of alkyl carbamates
fr_alkyl_halide	Number of alkyl halides
fr_allylic_oxid	Number of allylic oxidations
fr_amide	Number of amides
fr_amidine	Number of amidines
fr_aniline	Number of anilines
fr_aryl_methyl	Number of aryl methyl groups
fr_azide	Number of azides

Table A.1. continued. List and descriptions of RDKit descriptors

fr_azo	Number of azo groups
fr_barbitur	Number of barbituric acid derivatives
fr_benzene	Number of benzene rings
fr_benzodiazepine	Number of benzodiazepines
fr_bicyclic	Number of bicyclic rings
fr_diazo	Number of diazo groups
fr_dihydropyridine	Number of dihydropyridines
fr_epoxide	Number of epoxides
fr_ester	Number of esters
fr_ether	Number of ether groups
fr_furan	Number of furan rings
fr_guanido	Number of guanidine groups
fr_halogen	Number of halogens
fr_hdrzine	Number of hydrazines
fr_hdrzone	Number of hydrazones
fr_imidazole	Number of imidazole rings
fr_imide	Number of imides
fr_isocyan	Number of isocyanates
fr_isothiocyan	Number of isothiocyanates

Table A.1. continued. List and descriptions of RDKit descriptors

fr_ketone	Number of ketones
fr_ketone_Topliss	Number of ketones (Topliss modification)
fr_lactam	Number of lactams
fr_lactone	Number of lactones
fr_methoxy	Number of methoxy groups
fr_morpholine	Number of morpholine rings
fr_nitrile	Number of nitriles
fr_nitro	Number of nitro groups
fr_nitro_arom	Number of nitro groups attached to an aromatic ring
fr_nitro_arom_nonortho	Number of nitro groups attached to an aromatic ring, excluding those attached to ortho positions
fr_nitroso	Number of nitroso groups
fr_oxazole	Number of oxazole rings
fr_oxime	Number of oxime groups
fr_para_hydroxylation	Number of para-hydroxylations
fr_phenol	Number of phenols
fr_phenol_noOrthoHbond	Number of phenols not involved in ortho hydrogen bonds
fr_phos_acid	Number of phosphoric acid derivatives
fr_phos_ester	The number of phosphoester groups in the molecule

Table A.1. continued. List and descriptions of RDKit descriptors

fr_piperdine	The number of piperidine rings in the molecule
fr_piperzine	The number of piperazine rings in the molecule
fr_priamide	The number of primary amides in the molecule
fr_prisulfonamd	The number of primary sulfonamides in the molecule
fr_pyridine	The number of pyridine rings in the molecule
fr_quatN	The number of quaternary nitrogens in the molecule
fr_sulfide	The number of sulfide (-S-) linkages in the molecule
fr_sulfonamd	The number of sulfonamide (-S(=O)(=O)-N-) groups in the molecule
fr_sulfone	The number of sulfone (-S(=O)(=O)-) groups in the molecule
fr_term_acetylene	The number of terminal acetylenes in the molecule
fr_tetrazole	The number of tetrazole rings in the molecule
fr_thiazole	The number of thiazole rings in the molecule
fr_thiocyan	The number of thiocyanate (-SCN) groups in the molecule
fr_thiophene	The number of thiophene rings in the molecule
fr_unbrch_alkane	The number of unbranched alkanes in the molecule
fr_urea	The number of urea (-NH-CO-NH-) groups in the molecule
qed	The quantitative estimate of drug-likeness of the molecule

VITA

Nicholas R. Furth was born on March 8th, 1998, in Morristown, New Jersey on a dark and stormy night. He graduated from Morris County School of Technology in May of 2017. Nicholas attended the New Jersey Institute of Technology (Newark NJ) receiving Bachelor of Science and Master of Science degrees in May 2021 and May 2022 respectively. The following August he began attending the University of Tennessee as part of the Bredesen Center for Interdisciplinary research and Graduate Education under the advisement of Dr. Thomas A. Zawodzinski.